

Internal note: HEN-4xx

05 November 2003.

## PRELIMINARY

### The STOV and STSX firmware design

#### Abstract

The STOV (Scintillatingfiber Tracker OVerlay) boards and the STSX (Scintillating fiber Tracker SeXtant) boards are the last ones in the chain of data transport for the Central Tracking Trigger (CTT) data out of the D0 detector. The STOV board accepts data coming from 6 DF EA (Digital Front End Analog) boards, selects the L2CFT records and sends out a sorted list of tracks to 2 STSX boards that cover sectors adjacent to the overlap region. The STSX board accepts data coming from 7 or 8 DF EA boards and 2 STOV boards, selects the L2CFT input records and passes them on to the L2STT (Level 2 Trigger) and L3 (Level 3) boards.

The firmware for the Xilinx FPGAs is written in VHDL. Its function is to synchronize the incoming LVDS data, select the L2CFT records and store them in internal Dual Port Memories. Before sending to the output (STOV to STSX or STSX to L2STT), track descriptor words are sorted in order to bring tracks with high Pt values to the beginning of the output list. The L3 output record of the STSX boards provide a copy of all L2CFT input records for certification and diagnostics purposes.

Author: T.A.M. Wijnen, University of Nijmegen.

Keywords: D0, CTT, STOV, STSX, STT.

Version: 0.80

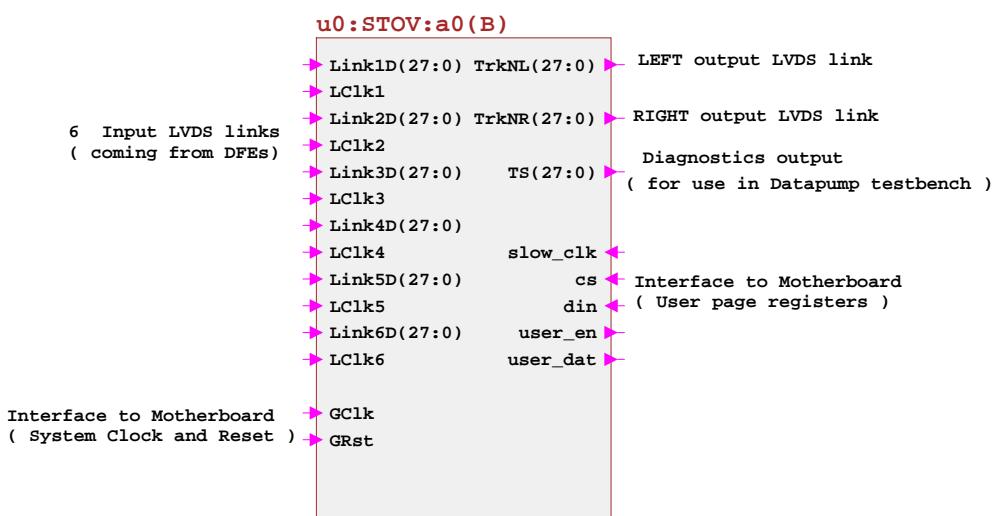
Date: 06 Nov. 2003.

## 1 Introduction

The Central Track Trigger (CTT) of the D0 Read out electronics (RUN II) is implemented on hundreds of FPGAs. Most of these reside on the daughter boards that are mounted on a common type of Digital Front End Motherboards (DFEM). Each of the FPGAs in the system receives data on several (maximum 10) LVDS serial data links and outputs its result on one or more LVDS links and/or two G-Links. The firmware takes care of storing the input records, selects and processes the data and forwards the result to the next level in the system.

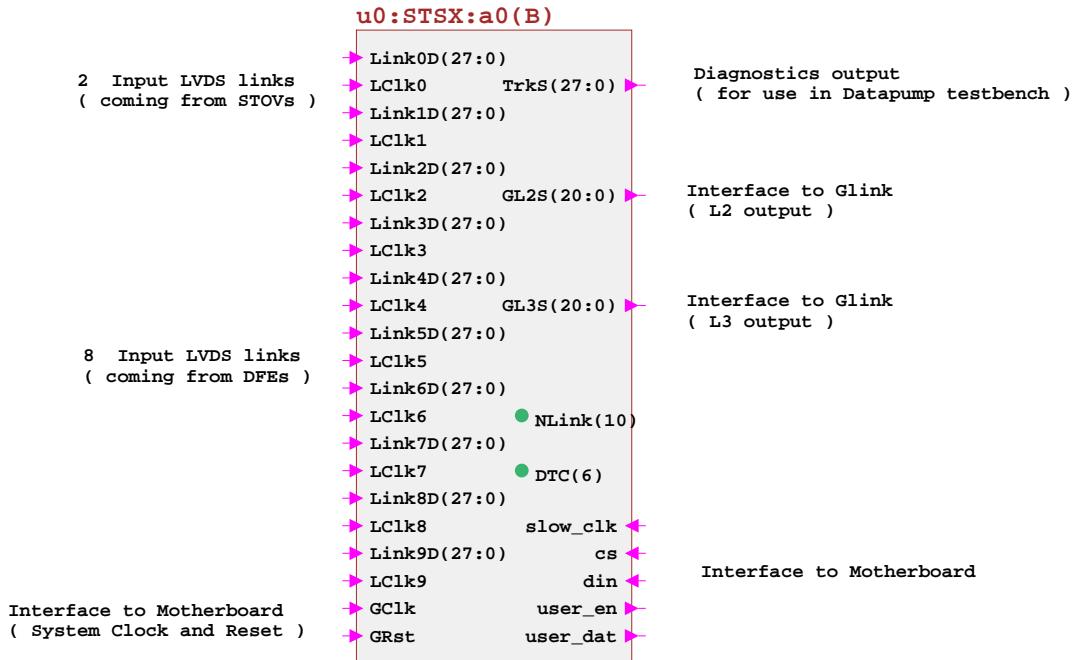
The STOV firmware is implemented in one of the three FPGAs that are mounted on a double-wide daughterboard; the two other FPGAs are unused. The tasks of the STOV firmware are:

1. Synchronize the input records and make the transition from data running on individual Link clocks to data running on the FPGA system (Global) clock.
2. Store the L2CFT records (i.e. a header, followed by a list of zero to a maximum of 24 track descriptor words and a trailer) in Dual Port Memories. This is needed because sorting may only start when all input receivers have received (at least) the header of the L2CFT record or recognised that this link was invalid as a result of being too late or absent.
3. Sort tracks on their Pt value such that tracks with the highest Pt will appear in the beginning of the output list.
4. Select tracks on the basis of Pt sign and sector number to go out on the “left”, the “right” or both outputs and send the final list to the STSX boards that cover the sectors on the left and right of the overlay region.



The STSX firmware is implemented in one of the three FPGAs that are mounted on a double-wide daughterboard; the two other FPGAs are unused. The tasks of the STSX firmware are:

1. Synchronize the input records and make the transition from data running on individual Link clocks to data running on the FPGA system (Global) clock.
2. Store the L2CFT records (i.e. a header, followed by a list of zero to a maximum of 24 track descriptor words and a trailer) in Dual Port Memories. This is needed because sorting may only start when all input receivers have received (at least) the header of the L2CFT record or recognised that this link was invalid as a result of being too late or absent.
3. Sort tracks on their Pt value such that tracks with the highest Pt will appear in the beginning of the output list.
4. Format the L2STT output record according to the defined protocol for the G-Link transmitter on the transition board.
5. Assemble a copy of all input records and format them according to the defined protocol for sending to the L3 output via another G-Link transmitter.



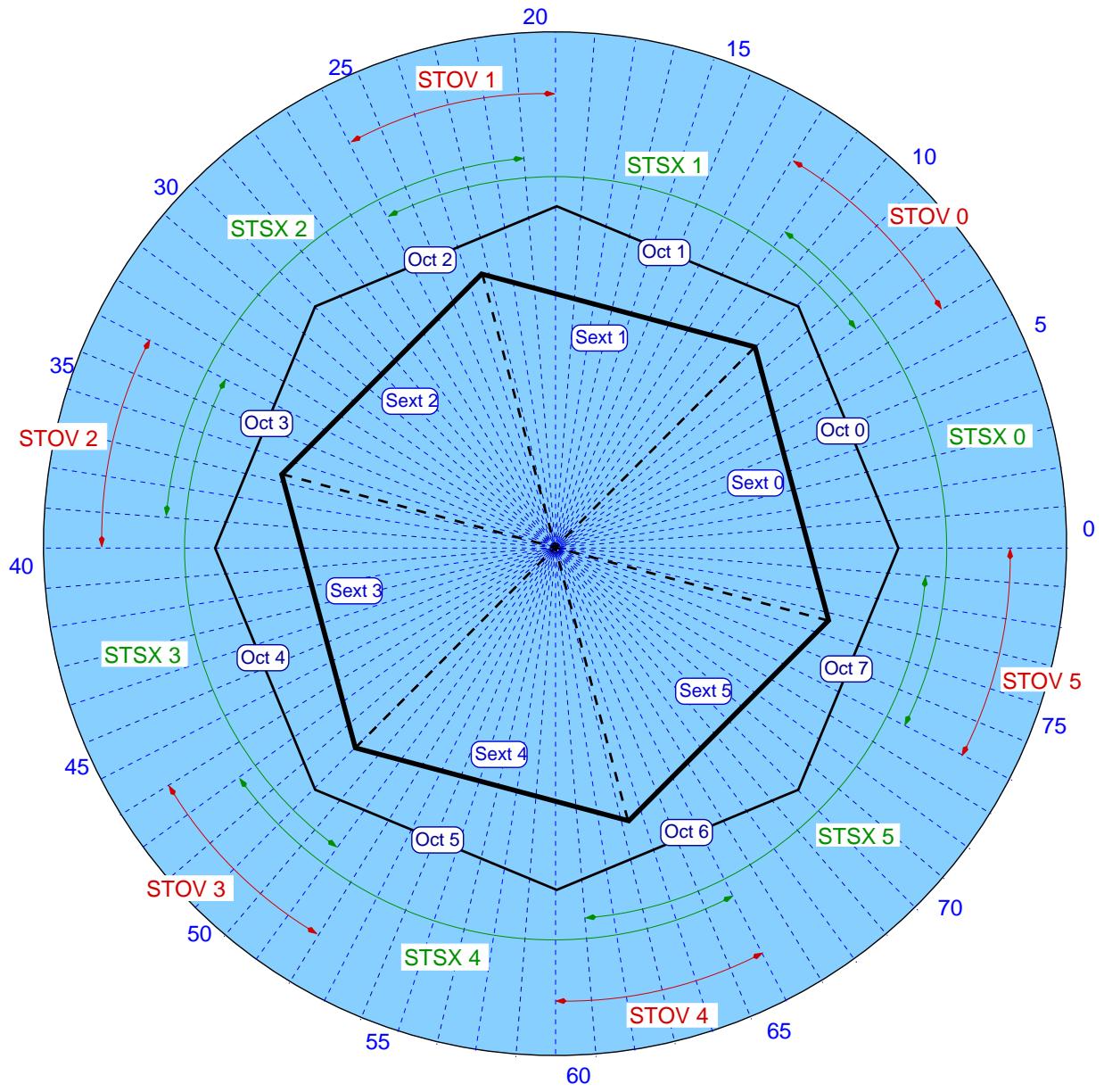


Figure 1: Overlap regions

## 2 Design organisation and parameters

The firmware for STOV and STSX will be loaded into a Xilinx Virtex FPGA, type XCV600BG560, located on the DWDB daughterboard of the commonly used DFE Motherboard. The FPGA has 10 input links, each of which consists of a serial LVDS link delivering 28 bit words using its own transmission clock, running at the accelerator RF clock, i.e. 53 MHz. Clocks across links can be skewed to several nanoseconds due to e.g. cable length variations. The start of a L2CFT data record may be skewed by 2 ticks due to differences between processing across DFEA boards. Due to the layout of the 80-fold detector front-end system, the STOV boards have 6 inputs and STSX boards have 7 or 8 DFEA inputs. All STSX boards have two STOV inputs, each delivering preprocessed L2CFT records of 6 DFEA boards. The STOV records arrive at the STSX some 25 clock ticks later than the records coming straight from the DFEAs.

Both the L2STT and L3 outputs drive a fiber optic link (G-Link [xxx]) via a 16 bit wide interface running off the Global Clock. The Global Clock for the FPGA and the GLinks is taken from the LVDS input Link #2; the Global Clock (GClk) will thus have a typical delay of 4 ns with respect to Link Clock #2. The maximum number of tracks in the L2CFT input record from DFEAs and STOVs will be 24. The output to L2STT will have a maximum of 48 tracks. The length of the L3 record is equal to the total of all 10 input records.

The VHDL designs are setup as projects in the EASE (Translogic [xx]) graphical design package. This allows us to graphically organise and interconnect VHDL entities with both hierarchical block type and plain text behavioural architectures. The full VHDL output file from the EASE project is then synthesized into logic with the Leonardo Spectrum package (Exemplar, [xx]) and the resulting EDIF file is implemented into Virtex FPGA with the Xilinx ISE package. Modelsim is the program used for the behavioural simulation of the functional design as well as the back-annotated design with its real timing.

## 3 Format of records

The format of the L2CFT records that the STOV and STSX should select out of the total stream of data that is running over the LVDS links is as given in Fig. 2. (A full description is given in the D0 CTT Protocols document [1])

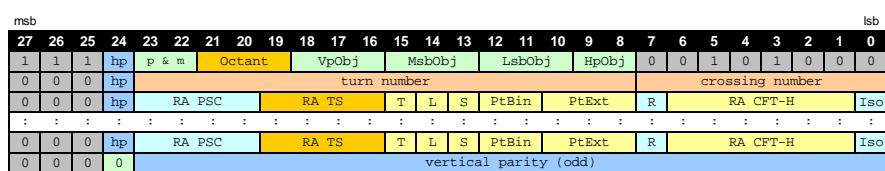


Figure 2: L2CFT record

The STOV has to decide which tracks should go to the left output (i.e. to the STSX on the left side of this overlap region), which tracks go to the right output and which tracks go to both. It set flags according to the selection criteria and stores them along with the

track descriptor in its memory. The format is shown in Fig. 3.

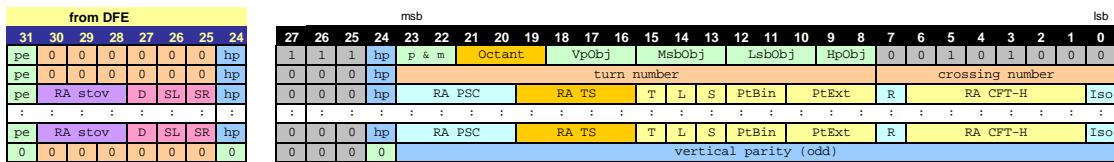
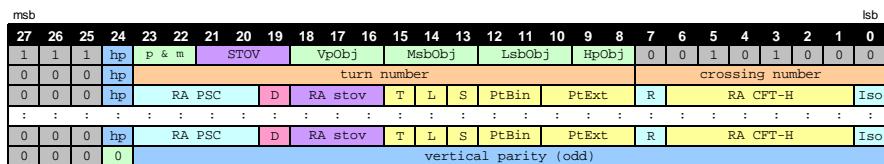
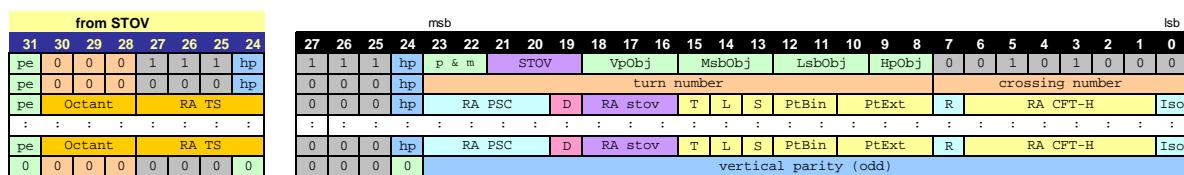


Figure 3: Stored record in the STOV

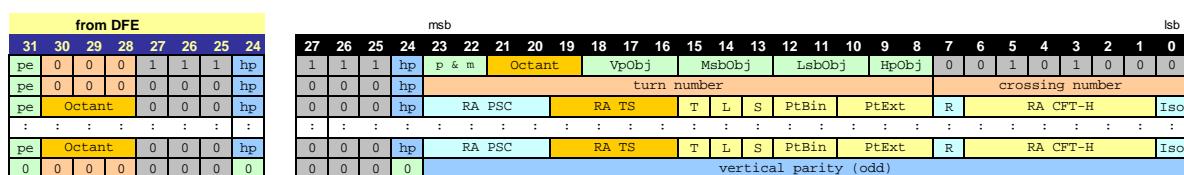
The STOV boards send records to the STSX that look like L2CFT records, but note that instead of sending the Octant number in the header, they send an STOV number. Also, in the track information word, the 4 bits trigger sector number has been replaced by a 3 bits relative wedge number and 1 “D” bit (when this track was sent to the left and right output of the STOV).



In order to construct the absolute trigger sector number in the output list of tracks going to L2STT, the STSX must first recalculate the trigger sector number and the octant number from wedge number and STOV number. It stores this information in the dual port memory using an extra 8 bits in addition to the 24 bits it received from the STOV.

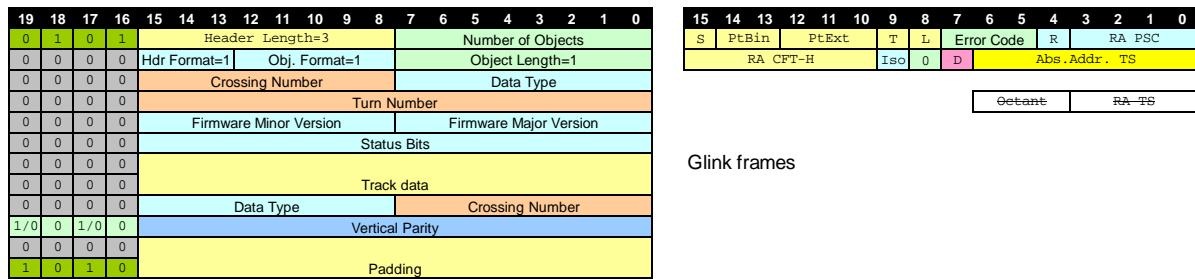


There is a similar problem with the records coming straight from the DFEAs, but here the STSX simply has to add the octant number from the header word to each of the individual track words and store that in the dual port memory. Thus, when it comes to the final sorting of tracks according to the highest Pt number, the full octant/relative sector number is available in each track word. The L2 Sender can than easily calculate an absolute trigger sector number and put that inside the outgoing records.

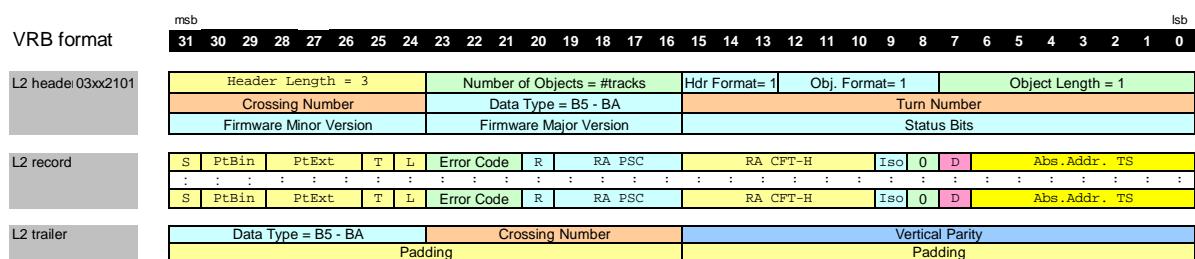


Here is how the full GLink record looks in 16 bit frames, how the control bits are added

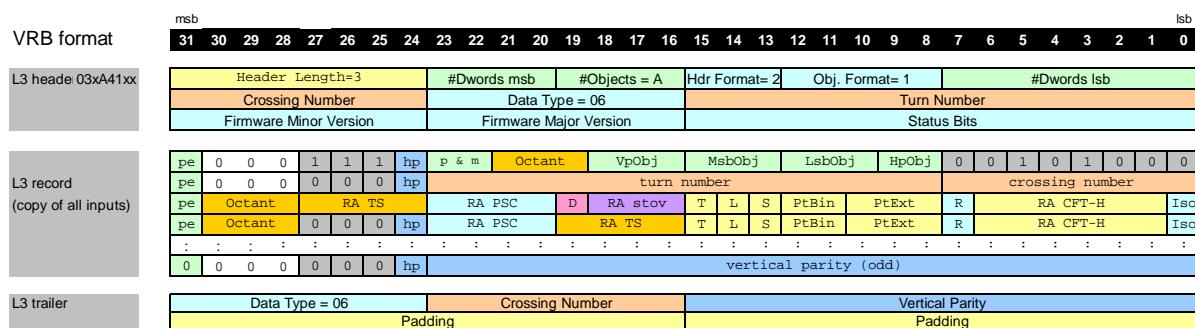
and how each track data word is reformatted.



Here are the resulting GLink records as they show up in the receiving VRB for L2STT. Note that the trigger sector number is now given in an absolute number (not as Octant and rel. sector number). The data type codes for STSX #0 ... STSX #5 are in hex numbers 0xB5 ... 0xBA.



Here are the resulting GLink records as they show up in the receiving VRB for L3. Note that trigger sector number for records coming straight from the DFEAs is now given as an octant number in the header and a relative sector number in the data. The records coming from the STOVs have a STOV number in the header and a relative wedge number in the track data. The data type codes for STSX #0 ... STSX #5 are the same hex number 0x06.



#### 4 STOV Overlay track selection

The code for the selector that determines whether a track should go to the left side STSX or the right side STSX or both, is a translation of the tables in the "Overlap" document

(March 26, 2001) by Bill Lee. Note that in the document tables, Ptbin "00" represents the highest Pt bin!

The VHDL code selects tracks on the basis of the octant/sextant geometry, wedge number (i.e. relative sector number in this octant) and the Pt Sign bit. This ensures that tracks going to both outputs, are restricted to be tracks coming from only 3 sectors in that overlap region. The VHDL code works for tracks in any of the DFE inputs in any of the STOV cards (all STOV FPGAs have the same code).

STOV	Octant	$P_t$ Sign	Sector Wedge	$\Phi / P_t$ bin	Both	Left	Right	Left STSX	Right STSX
0	0	0	$w \geq 9$	all	yes	yes	yes	1	0
		0	$w < 9$	all	no	no	yes		
		1	$w \geq 8$	all	yes	yes	yes		
		1	$w < 8$	all	no	no	yes		
0	1	0	$w \leq 1$	all	yes	yes	yes	1	0
		0	$w > 1$	all	no	yes	no		
		1	$w \leq 0$	all	yes	yes	yes		
		1	$w > 0$	all	no	yes	no		

STOV	Octant	$P_t$ Sign	Sector Wedge	$\Phi / P_t$ bin	Both	Left	Right	Left STSX	Right STSX
1	2	0	$w \geq 2$	all	no	yes	no	2	1
		0	$w \leq 5$	all	no	no	yes		
		0	$2 \leq w \leq 5$	all	yes	yes	yes		
		1	$w \geq 1$	all	no	yes	no		
		1	$w \leq 4$	all	no	no	yes		
		1	$1 \leq w \leq 4$	all	yes	yes	yes		
2	3	0	$w \geq 5$	all	no	yes	no	3	2
		0	$w \leq 8$	all	no	no	yes		
		0	$5 \leq w \leq 8$	all	yes	yes	yes		
		1	$w \geq 4$	all	no	yes	no		
		1	$w \leq 7$	all	no	no	yes		
		1	$4 \leq w \leq 7$	all	yes	yes	yes		

The tracks in the octants 4 ... 7 follow the same rules as the tracks in octants 0 ... 3.

STOV	Octant	$P_t$	Sector	$\Phi / P_t$	bin	Both	Left	Right	Left STSX	Right STSX
3	4	0	$w \geq 9$	all	yes	yes	yes	yes	4	3
		0	$w < 9$	all	no	no	yes	yes		
		1	$w \geq 8$	all	yes	yes	yes	yes		
		1	$w < 8$	all	no	no	yes	yes		
3	5	0	$w \leq 1$	all	yes	yes	yes	yes	4	3
		0	$w > 1$	all	no	yes	no	no		
		1	$w \leq 0$	all	yes	yes	yes	yes		
		1	$w > 0$	all	no	yes	no	no		
4	6	0	$w \geq 2$	all	no	yes	no	no	5	4
		0	$w \leq 5$	all	no	no	yes	yes		
		0	$2 \leq w \leq 5$	all	yes	yes	yes	yes		
		1	$w \geq 1$	all	no	yes	no	no		
		1	$w \leq 4$	all	no	no	yes	yes		
		1	$1 \leq w \leq 4$	all	yes	yes	yes	yes		
5	7	0	$w \geq 5$	all	no	yes	no	no	0	5
		0	$w \leq 8$	all	no	no	yes	yes		
		0	$5 \leq w \leq 8$	all	yes	yes	yes	yes		
		1	$w \geq 4$	all	no	yes	no	no		
		1	$w \leq 7$	all	no	no	yes	yes		
		1	$4 \leq w \leq 7$	all	yes	yes	yes	yes		

## 5 L2STT Testvectors

When put in testmode, the STSX will await the header of a L2CFT record on Link #2, copy the crossing and turn numbers from the input link and send out the records with the fake tracks from the list below.

The testvectors are different for each STSX. They are extracted from the list that I received from Ulrich Heintz (09-May-2003). The frames are shown in 20-bit data as they sent to the GLINK transmitter. The tracks are ordered by Pt value. Each event will have different crossing and turn numbers.

### 5.1 STSX #0 Testvectors

frame1	frame2	STT receiver
50308 hdr-len 03 n-obj 08	02101 hdr-fmt 21 obj-len 01	0x03082101
004B5 X-num 04 dat-typ B5	00000 turn-num 0000	0x04B50000
00001 firm-num 0001	00100 status: 0100	0x00010100
frame1 S PB PE T L E R Psc	frame2 CFT I D ATS=OR -> track:	
00C00 0 0 3 0 0 0 0 0	03C00 0F 0 0 00 00 -> 000031E	0x0C003C00

02000	0	1	0	0	0	0	0	03C00	0F	0	0	00	00	->	000081E	0x20003C00	
04400	0	2	1	0	0	0	0	0A401	29	0	0	01	01	->	0011152	0x4400A401	
06000	0	3	0	0	0	0	0	0AC02	2B	0	0	02	02	->	0021856	0x6000AC02	
0E209	1	3	0	1	0	0	0	9	0AC02	2B	0	0	02	02	->	092B856	0xE209AC02
0E219	1	3	0	1	0	0	1	9	00003	00	0	0	03	03	->	093B880	0xE2190003
06400	0	3	1	0	0	0	0	00003	00	0	0	03	03	->	0031900	0x64000003	
07400	0	3	5	0	0	0	0	03400	0D	0	0	00	00	->	0001D1A	0x74003400	
0B504	-	-	-	-	-	-	-	A1AA8	end record.							0xB5041AA8	

## 5.2 STSX #1 Testvectors

frame1	frame2	STT receiver
50308 hdr-len 03 n-obj 08	02101 hdr-fmt 21 obj-len 01	0x03082101
008B6 X-num 08 dat-typ B6	00000 turn-num 0000	0x08B60000
00001 firm-num 0001	00100 status: 0100	0x00010100
frame1 S PB PE T L E R Psc	frame2 CFT I D ATS=OR -> track:	
02400 0 1 1 0 0 0 0 0	0288A 0A 0 1 0A 10 -> 9000914	0x2400288A
02400 0 1 1 0 0 0 0 0	0980C 26 0 0 0C 12 -> 102094C	0x2400980C
02800 0 1 2 0 0 0 0 0	02C8A 0B 0 1 0A 10 -> 9000A16	0x28002C8A
02800 0 1 2 0 0 0 0 0	0288A 0A 0 1 0A 10 -> 9000A14	0x2800288A
06000 0 3 0 0 0 0 0 0	0740B 1D 0 0 0B 11 -> 101183A	0x6000740B
06800 0 3 2 0 0 0 0 0	0288A 0A 0 1 0A 10 -> 9001A14	0x6800288A
0E800 1 3 2 0 0 0 0 0	0040D 01 0 0 0D 13 -> 1033A02	0xE800040D
06C00 0 3 3 0 0 0 0 0	02C8A 0B 0 1 0A 10 -> 9001B16	0x6C002C8A
0B608 - - - - - -	AD136 end record. (trailer)	0xB608D136

## 5.3 STSX #2 Testvectors

frame1	frame2	STT receiver
5030C hdr-len 03 n-obj 0C	02101 hdr-fmt 21 obj-len 01	0x030C2101
017B7 X-num 17 dat-typ B7	00000 turn-num 0000	0x17B70000
00001 firm-num 0001	00100 status: 0100	0x00010100
frame1 S PB PE T L E R Psc	frame2 CFT I D ATS=OR -> track:	
02E06 0 1 3 1 0 0 0 6	04CA4 13 0 1 24 36 -> B668B26	0x2E064CA4
02E06 0 1 3 1 0 0 0 6	048A4 12 0 1 24 36 -> B668B24	0x2E0648A4
04207 0 2 0 1 0 0 0 7	02022 08 0 0 22 34 -> 3749010	0x42072022
04206 0 2 0 1 0 0 0 6	04CA4 13 0 1 24 36 -> B669026	0x42064CA4
04606 0 2 1 1 0 0 0 6	04CA4 13 0 1 24 36 -> B669126	0x46064CA4
04606 0 2 1 1 0 0 0 6	048A4 12 0 1 24 36 -> B669124	0x460648A4
0C60A 1 2 1 1 0 0 0 A	0A8A4 2A 0 1 24 36 -> BA6B154	0xC60AA8A4
0CA0A 1 2 2 1 0 0 0 A	0AOA4 28 0 1 24 36 -> BA6B250	0xCA0AA0A4
0C800 1 2 2 0 0 0 0 0	02A1D 0A 1 0 1D 29 -> 2093215	0xC8002A1D
0CA0D 1 2 2 1 0 0 0 D	09C22 27 0 0 22 34 -> 3D4B24E	0xCA0D9C22
04E12 0 2 3 1 0 0 1 2	094A4 25 0 1 24 36 -> B2693CA	0x4E1294A4
0E800 1 3 2 0 0 0 0 0	04022 10 0 0 22 34 -> 3043A20	0xE8004022
0B717 - - - - - -	A2D8D end record. (trailer)	0xB7172D8D

## 5.4 STSX #3 Testvectors

frame1	frame2	STT receiver
50305 hdr-len 03 n-obj 05 00EB8 X-num 0E dat-typ B8 00001 firm-num 0001	02101 hdr-fmt 21 obj-len 01 00000 turn-num 0000 00100 status: 0100	0x03052101 0x0EB80000 0x00010100
frame1 S PB PE T L E R Psc	frame2 CFT I D ATS=OR -> track:	
0260E 0 1 1 1 0 0 0 E	094A4 25 0 1 24 36 -> BE6894A	0x260E94A4
02C00 0 1 3 0 0 0 0 0	02E2E 0B 1 0 2E 46 -> 4060B17	0x2C002E2E
OE605 1 3 1 1 0 0 0 5	08029 20 0 0 29 41 -> 451B940	0xE6058029
OE605 1 3 1 1 0 0 0 5	07C29 1F 0 0 29 41 -> 451B93E	0xE6057C29
06A15 0 3 2 1 0 0 1 5	098A3 26 0 1 23 35 -> B559ACC	0x6A1598A3
OB80E - - - - - - -	02B81 trailer	0xB80E2B81
00000 0 0 0 0 0 0 0 0	00000 00 0 0 00 00 -> 00000000	0x00000000
00000 0 0 0 0 0 0 0 0	00000 00 0 0 00 00 -> 00000000	0x00000000
00000 0 0 0 0 0 0 0 0	A0000 end record.	0x00000000

## 5.5 STSX #4 Testvectors

frame1	frame2	STT receiver
50305 hdr-len 03 n-obj 05 021B9 X-num 21 dat-typ B9 00001 firm-num 0001	02101 hdr-fmt 21 obj-len 01 00000 turn-num 0000 00100 status: 0100	0x03052101 0x21B90000 0x00010100
frame1 S PB PE T L E R Psc	frame2 CFT I D ATS=OR -> track:	
04E07 0 2 3 1 0 0 0 7	04C39 13 0 0 39 57 -> 5779326	0x4E074C39
0620E 0 3 0 1 0 0 0 E	04C39 13 0 0 39 57 -> 5E79826	0x620E4C39
0720E 0 3 4 1 0 0 0 E	07439 1D 0 0 39 57 -> 5E79C3A	0x720E7439
0760E 0 3 5 1 0 0 0 E	07439 1D 0 0 39 57 -> 5E79D3A	0x760E7439
07400 0 3 5 0 0 0 0 0	07237 1C 1 0 37 55 -> 5051D39	0x74007237
OB921 - - - - - - -	095A3 trailer ->	0xB92195A3
00000 0 0 0 0 0 0 0 0	00000 00 0 0 00 00 -> 00000000	0x00000000
00000 0 0 0 0 0 0 0 0	00000 00 0 0 00 00 -> 00000000	0x00000000
00000 0 0 0 0 0 0 0 0	A0000 end record.	0x00000000

## 5.6 STSX #5 Testvectors

frame1	frame2	STT receiver
50302 hdr-len 03 n-obj 02 001BA X-num 01 dat-typ BA 00001 firm-num 0001	02101 hdr-fmt 21 obj-len 01 00000 turn-num 0000 00100 status: 0100	0x03022101 0x01BA0000 0x00010100
frame1 S PB PE T L E R Psc	frame2 CFT I D ATS=OR -> track:	
0E800 1 3 2 0 0 0 0 0	00248 00 1 0 48 72 -> 7023A01	0xE8000248
0F400 1 3 5 0 0 0 0 0	00A42 02 1 0 42 66 -> 6063D05	0xF4000A42
0BA01 - - - - - - -	08CB3 trailer ->	0xBA018CB3
00000 0 0 0 0 0 0 0 0	00000 00 0 0 00 00 -> 00000000	0x00000000
00000 0 0 0 0 0 0 0 0	A0000 end record.	0x00000000

## 6 Overview of in-/output records and internal memory formats

## L2 DFEA → STOV / STSX frame format

### **Stored information in STOV, available for transfer to STSX**

## STOV → STSX frame format

**Stored information in STSX, available for L2 and L3 output transfers**

Sorter-stov

Sorter-dfe

## Glink frames

L3 header 0x3A1xx	Header Length=3	#Dwords msb	#Objects = A	Hdr Format= 2	Obj. Format= 1	#Dwords lsb																
	Crossing Number	Data Type = 06		Turn Number																		
	Firmware Minor Version	Firmware Major Version		Status Bits																		
L3 record (copy of all inputs)	pe	0	0	0	1	1	1	hp	p & m	Octant	VpObj	MsbObj	LsbObj	HpObj	0	0	1	0	1	0	0	0
	pe	0	0	0	0	0	0	hp				turn number				crossing number						
	pe	Octant	RA	TS	RA	PSC	D	RA	stoy	T	L	S	PtBin	PtExt	R	RA CFT-H		Iso				
	pe	Octant	0	0	0	hp	RA	PSC	RA	TS	T	L	S	PtBin	PtExt	R	RA CFT-H		Iso			
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	
L3 trailer	0	0	0	0	0	0	hp			12			vertical parity (odd)									
	Data Type = 06			Crossing Number			Vertical Parity			Data Type = 04			Vertical Parity			Data Type = 04						

stov for links 0...1  
octant for links 2...9

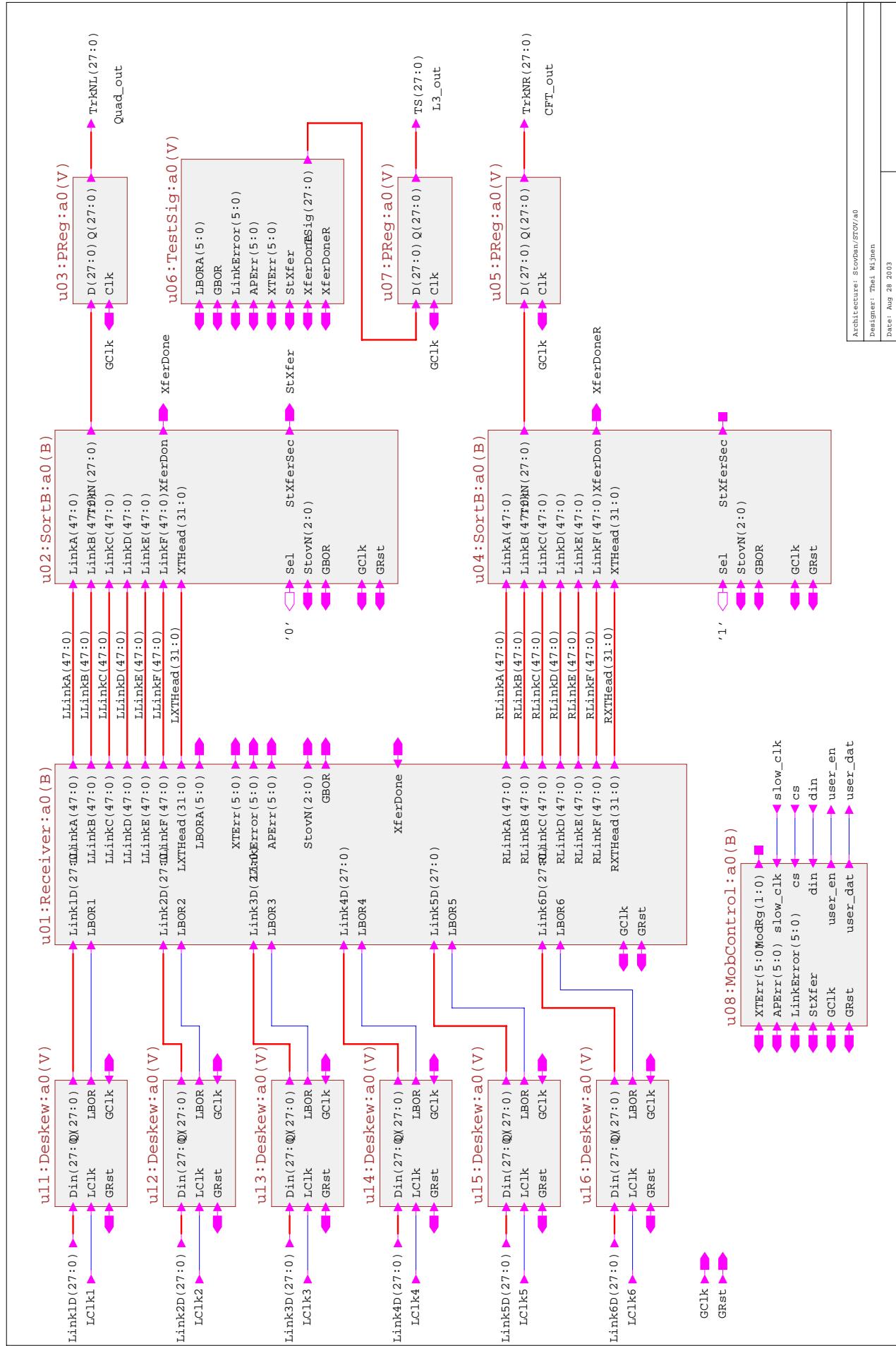
tracks for links 0...1  
tracks for links 2...9

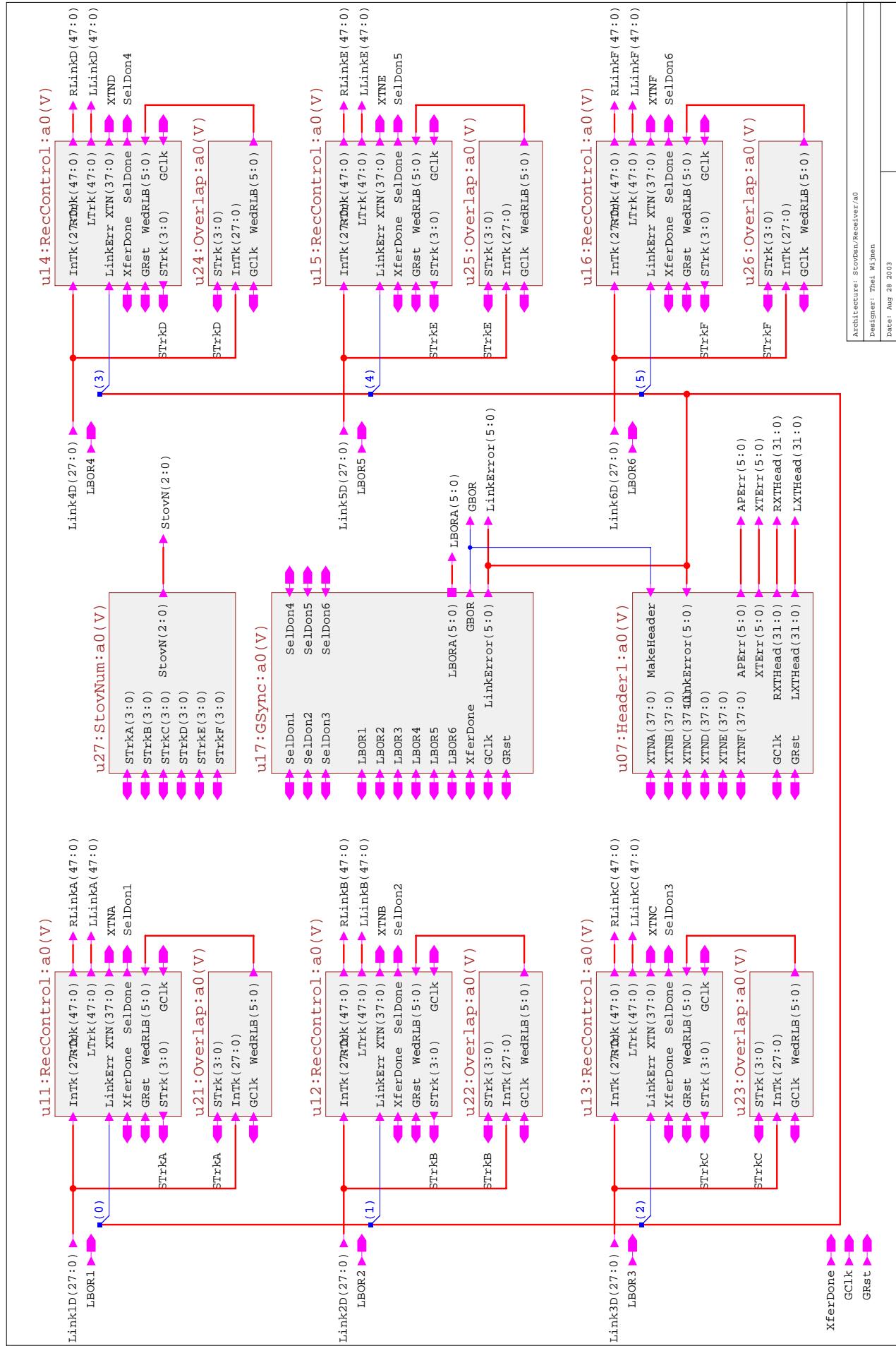
## 7 STOV VHDL Entities and components

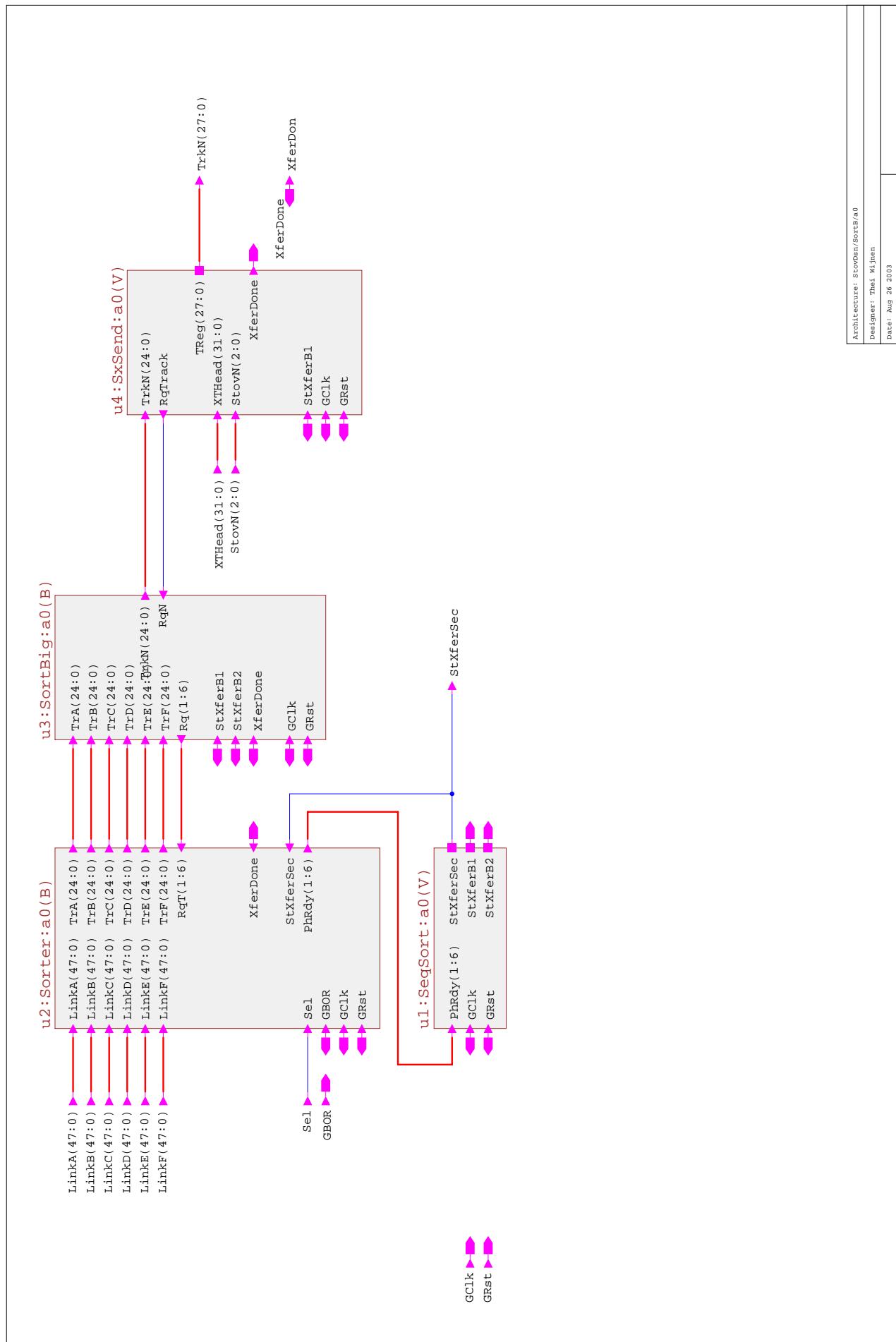
In the following paragraphs a short description is given of all VHDL entities that make up the STOV design. We will try and follow a top-down path through all hierarchical levels of the design. Each entity will be named according its hierarchical position in the top of the STOV design and marked with "(B)" when it has a block type structure or "(V)" when it has a behavioural (plain text) VHDL architecture.

## 8 STOV Design schematics

On the following pages the schematics for each of the block entities in the STOV design are shown.



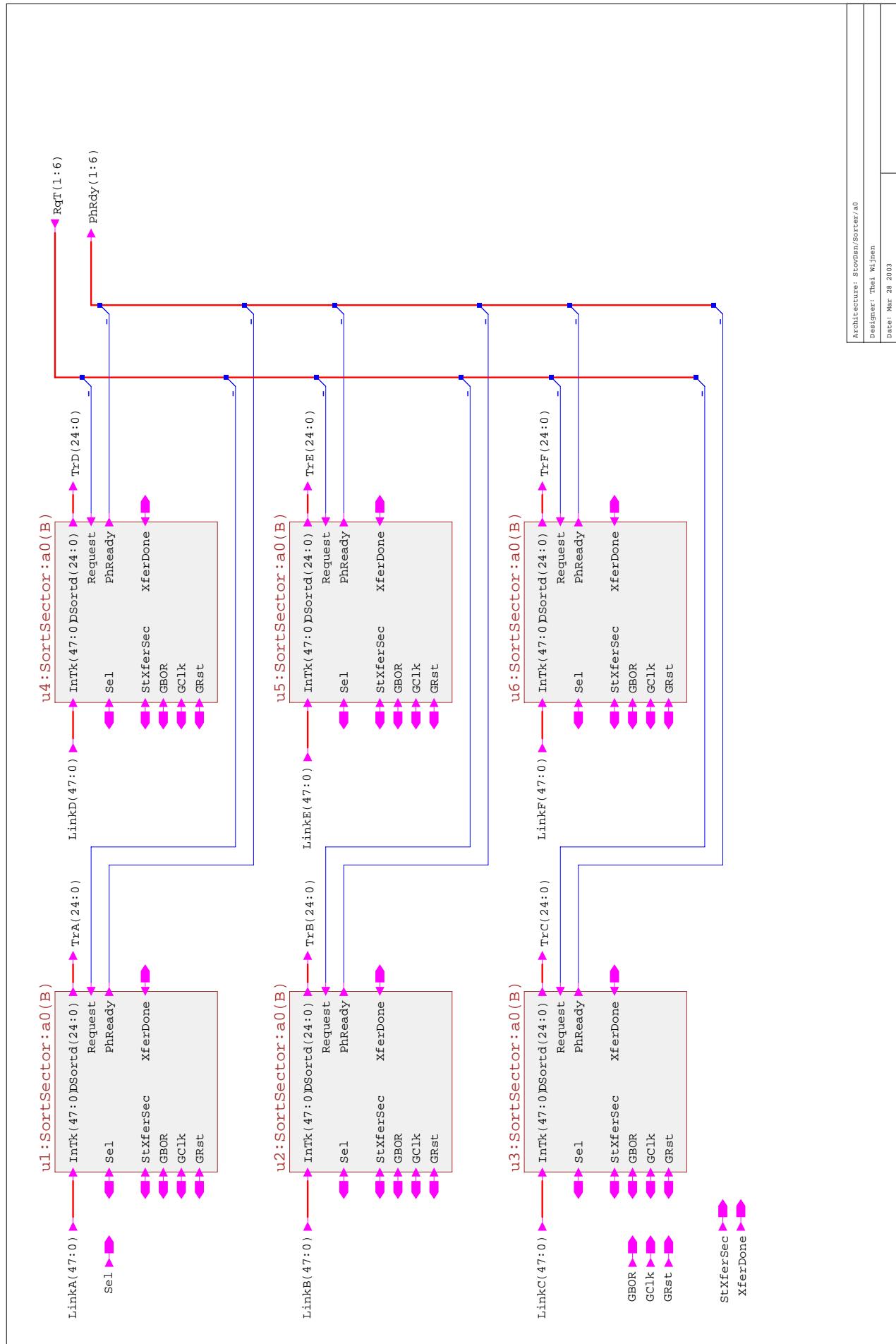


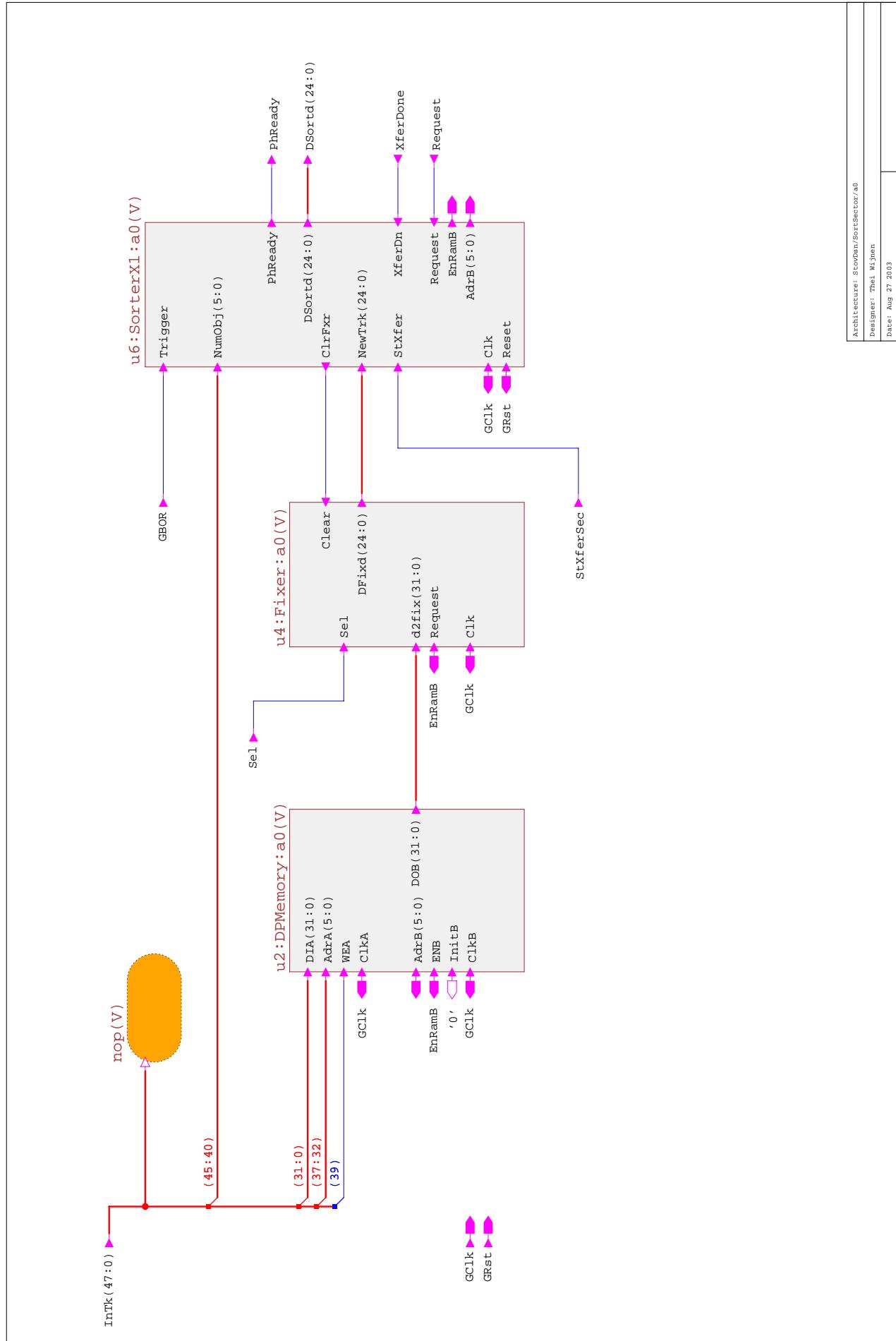


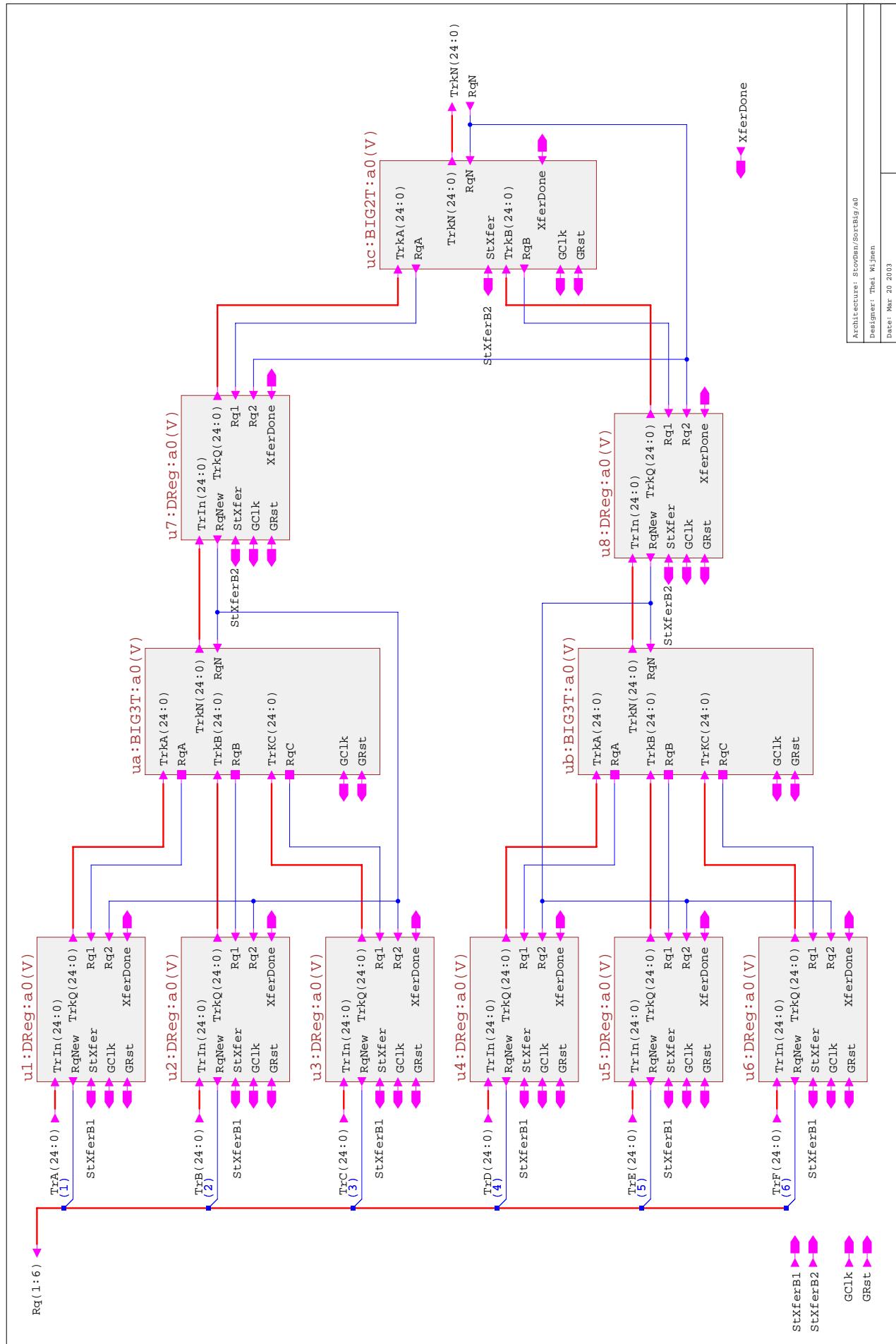
Architecture: Stovden/SortB/a0

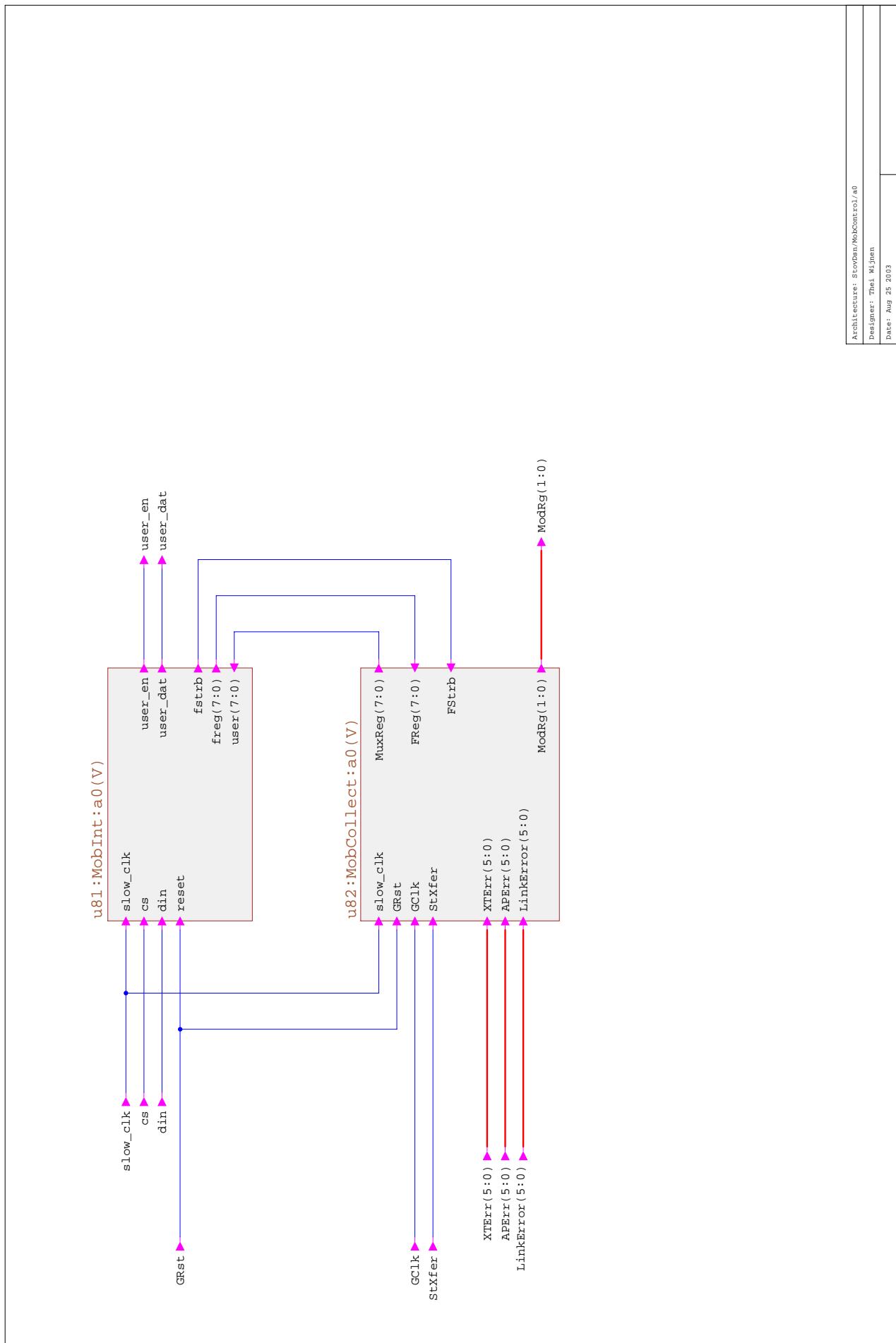
Designer: Theo Wijnen

date: Aug 26 2003









## 9 STSX VHDL Entities and components

In the following paragraphs a short description is given of all VHDL entities that make up the STSX design. We will try and follow a top-down path through all hierarchical levels of the design. Each entity will be named according its hierarchical position in the top of the STSX design and marked with "(B)" when it has a block type structure or "(V)" when it has a behavioural (plain text) VHDL architecture.

### 9.1 PReg (V)

This entity is a simple 28 bit (Link Clock driven) register that copies the input (LVDS) Link data from the FPGA input pins.

### 9.2 GReg (V)

This entity is a simple 21 bit (Global Clock driven) register that copies the output record data (16 bit data plus 5 control bits) to the FPGA output pins, driving the off-board G-Link interface.

### 9.3 GMux (V)

This entity is a 21 bit (Global Clock driven) multiplexer and register that copies the L2, the L3 or the combined L2 and L3 output record data (16 bit data plus 5 control bits) to the FPGA output pins, driving the off-board G-Link interface. When the two records are combined, it will take care that the total record contains just one header and one trailer frame with the special control bits set.

### 9.4 RecDFE (B)

This block entity contains the receivers, memories and sorters for the input records coming straight from the DFEAs. It also does the synchronisation of the records and checking of the header words. It consists of the following entities:

- |          |  |
|----------|--|
| Sorter8  | (B) The Sorter8 block entity contains the eight receivers and sorters for the tracks coming from the DFEA input links.   |
| SortBig3 | (B) This block entity contains a 3-to-1 track sorter, based on the Pt value of the track word.   |
| SortBig2 | (B) This block entity contains a 2-to-1 track sorter, based on the Pt value of the track word.   |
| SeqSort8 | (V) This entity controls the sequence of start signals for each of the sorter levels and the start of the L2 sender. All start signals are derived from the combination of the two StovRdy signals that come when each receiver for the STOV record has accepted a legal record. |

**Sorter8 (B)**

The Sorter8 block entity contains 8 RecSector blocks and Gsync8 and HeaderDfe.

- |           |  |
|-----------|--|
| RecSector | (B) The RecSector entity contains the receiver, memory and sorter for tracks coming from one sector or one DFEA input link.  |
| GSync8    | (V) This entity generates the Global Begin Of Record signal on the reception of the Begin of Record signal coming from Link #2. It also generates link error flags when the L2CFT record in a link is late or absent. It uses the octant numbers found in the header to determine which STSX board this one must be.           |
| HeaderDfe | (V) This entity checks the crossing and turn number in the header of the L2CFT record of Link #2 versus the number found in each of the other links. Any differences are flagged in XTErr flags. It also collects the parity error flags and it sets this flag when one or more frames of the input record had a parity error. |

**RecSector (B)**

The RecSector entity contains the receiver, memory and sorter for tracks coming from one sector or one DFEA input link. it consists of the following VHDL-text entities:

- |           |  |
|-----------|--|
| RecConDfe | (V) This entity looks for the beginning of the record, interprets the information in the header, calculates the vertical and horizontal parity, stores the tracks in the RAM, and counts the number of tracks received.  |
| DPMemory  | (V) The Dual Port Memory - implemented in a virtex Block RAM - uses independant address and clock signals to write only to port A and to read only from port B.  |
| SorterX1  | (V) This entity used to pre-sort 6 tracks assuming that they were not sorted within each PtBin. The sorting has been removed because it is already done in the DFEAs. Now this entity reads the tracks from the DPM every time there is a request from the 10-to-1 sorters behind this entity. |
| FixDFE    | (V) The sorters on Pt are based on all positive numbers for PtBin and PtExt So one must invert the incoming (negated definition) bits for PtBin/PtExt. The inversion must be undone in the output sender !   |
| DPMux     | (V) This entity multiplexes the address lines for port B of the DPM between the address driver coming from SorterX1 (for L2 output) and the address driver coming from the L3Sender. The signals EnS and L2disable control the multiplexer.  |

**SortBig3 (B) and SortBig2 (B)**

This block entity contains a 3-to-1 track sorter or a 2-to-1 track sorter, based on the Pt value of the track word. It contains the following VHDL-text entities:

- BIG3T (V) On every clock cycle this entity looks if there is a request for the next track. If RqN is equal to '1' it should select the track with the highest Pt from the three inputs. If RqA = '1' then the track with the highest Pt is in port "TrkA". If RqB = '1', then the highest Pt track is in port "TrkB", otherwise the highest Pt will be in port "TrkC". At the same time it enables the request line that goes to the entity from which the highest Pt track comes.
- BIG2T (V) This block entity contains a 2-to-1 track sorter, based on the Pt value of the track word. In normal (Running) mode this entity will on every clock cycle be looking whether there is a request for the next track (RqN = '1'). If so, then the highest Pt track is in port "TrkA" when AhigherB = '1'; else the highest Pt track is in port "TrkB" when BhigherA = '1'.
- BReg (V) This module is used to pipeline the choosing between 10 inputs. Given the way that the searching for the highest Pt track is done, it is not possible to pipeline it by just using a register (as usual). The Break module has two registers, register A keeps the latest tracks that has been received and register B stores a backup track that takes the place of A when that one has been sent.

## 9.5 RecSTOV (B)

This block entity contains the receivers, memories and sorters for the input records coming from the STOVs and it sorts and merges them with those of all DFEAs. Since the STOV records are the last ones to arrive, care is taken that the L2STT output sender derives its start signal from the STOV receivers (or on a timeout). The RecStov block entity contains the following entities: RecStovSec, HeaderSt, GSync2 and SortBig3.

- RecStovSec (B) This block entity contains the receivers, memories and sorters for tracks coming from one STOV input link. It is built from the entities RecConStov, DPMemory, FixOct SorterX1, DPMux.
- RecStovSec (V) This receiver module is the one that first take the row information coming into the chip, this module looks for the beginning of the record, interprets the information in the header, calculates the vertical and horizontal parity, stores the tracks in the RAM, and counts the number of tracks received for every of the 4 possible Pt bins.
- FixOct (V) The sorters on Pt are based all positive numbers for PtBin and PtExt So one must invert the incoming (negated definition) bits for PtBin. The inversion must be undone in the output sender !
- SorterX1 (V) see above
- DPMemory (V) see above
- DPMux (V) see above
- HeaderSt (V) This entity checks the individual crossing and turn number versus the “sacred” one coming from Link #2. It sets the appropriate flags when this is not the case and it calculates the total number of tracks.
- GSync2 (V) This entity generates Global Begin Of Record on the reception of the Begin of Record signal coming from Link #2 after a fair amount of delay. It also generates linkerror flags when the L2CFT record in a link is late or absent. It uses the octant numbers found in the header to determine which STSX board this one must be.
- SortBig3 (B) This block entity contains a 3-to-1 track sorter, based on the Pt value of the track word. (see above)

## 9.6 L2Send (B)

This block entity contains the reformatter and driver for the L2STT output G-Link. It adds the special G-Link header and trailer words. The L2Send block entity contains the following entities: SxSend, DPMemL2, DeMux and RegFile.

- |         |   |
|---------|---|
| SxSend  | (V) This entity fetches the tracks from the sorters, inverses the Pt value and creates the cctant number that is attached to each track. Then the list of tracks is stored in a Dual Port Memory. The extra memory decouples the sorting and fetching from the actual sender which is useful because the GLink sender can only send one track in two clock cycles because it sends 16 bits per clock cycle.                                   |
| DPMemL2 | (V) This is the actual dual port memory which is implemented in Xilinx Block RAMs. The memory is 128 words deep; only 48 tracks will be stored as a maximum. The upper half of the memory is used to store the L2STT testvectors (the numbers are embedded in the firmware). These testvectors can be sent to L2STT - instead of the real tracks - when the appropriate bit has been set in a control register via the motherboard interface. |
| DeMux   | (V) This entity demultiplexes the tracks coming from the memory (and the header) into two 16 bit word halves, before sending this data to the GLink transmitter. It also adds the trailer (vertical parity) word and does zero padding to insure that the record has a total word count that is a multiple of 8. Moreover it also takes care that - when enabled - the L2 record is sent out via the L3 output.                               |
| RegFile | (V) This entity generates the six word header and the 1 word trailer for the outgoing L2 record and feeds it into DeMux.  |

## 9.7 L3Merge (V)

This entity simply takes a number of signals coming from the front end parts, stores them and preformats them such that they can be fed to the “standard” L3Sender that was first designed by Satish Desai.

## 9.8 MobControl (B)

This block entity controls the interface to the motherboard. It consists of:

- |            |   |
|------------|---|
| MobInt     | (V) This is “standard” entity (written by J.Olsen) that handles the interface signals with the motherboard.   |
| MobCollect | (V) This entity gathers the error flags and puts them in 8 bit user page registers that the motherboard can read. It also holds a register via which the STSX output modes may be controlled, e.g. whether the L2STT record will also be sent to the L3 output, |

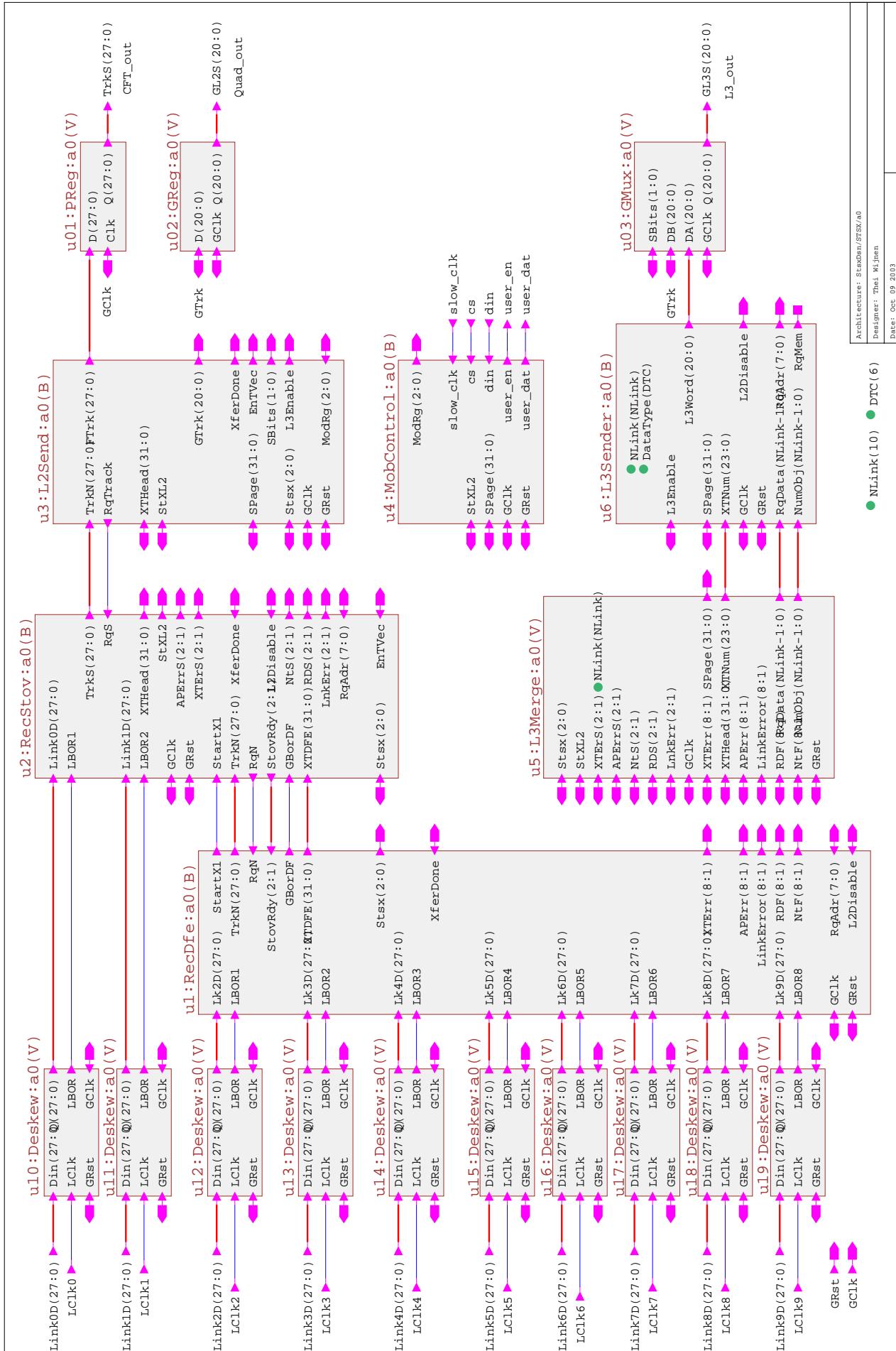
## 9.9 L3Sender (B)

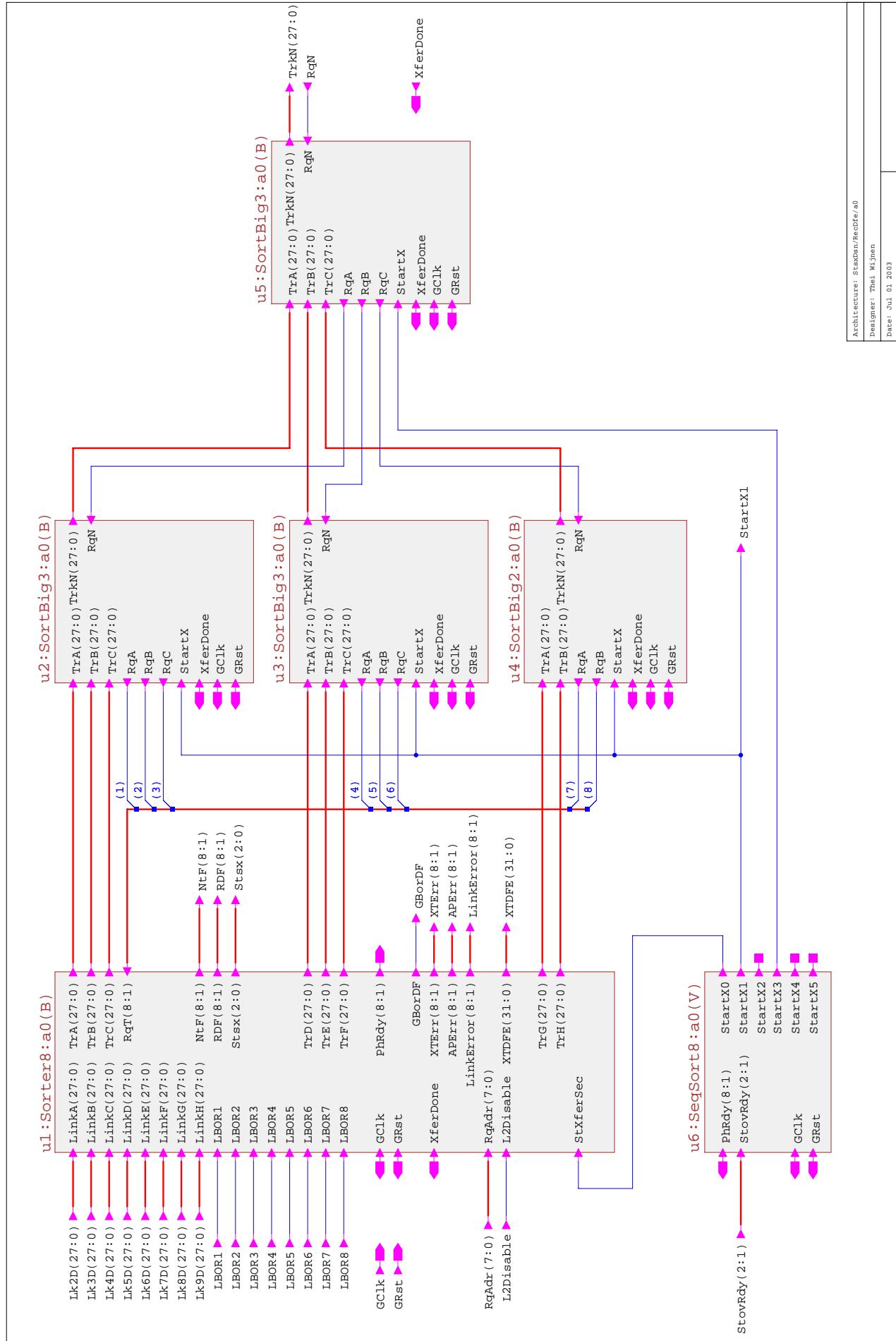
This block entity formats and sends the L3 record and sends it (via GMux) to the offboard Glink transmitter. It gets the data, number of tracks and the flags for the header from L3Merge and drives the memories inside the entities RecDfe and RecStov to retrieve a copy of the input L2CFT records. The L3Sender block consists of:

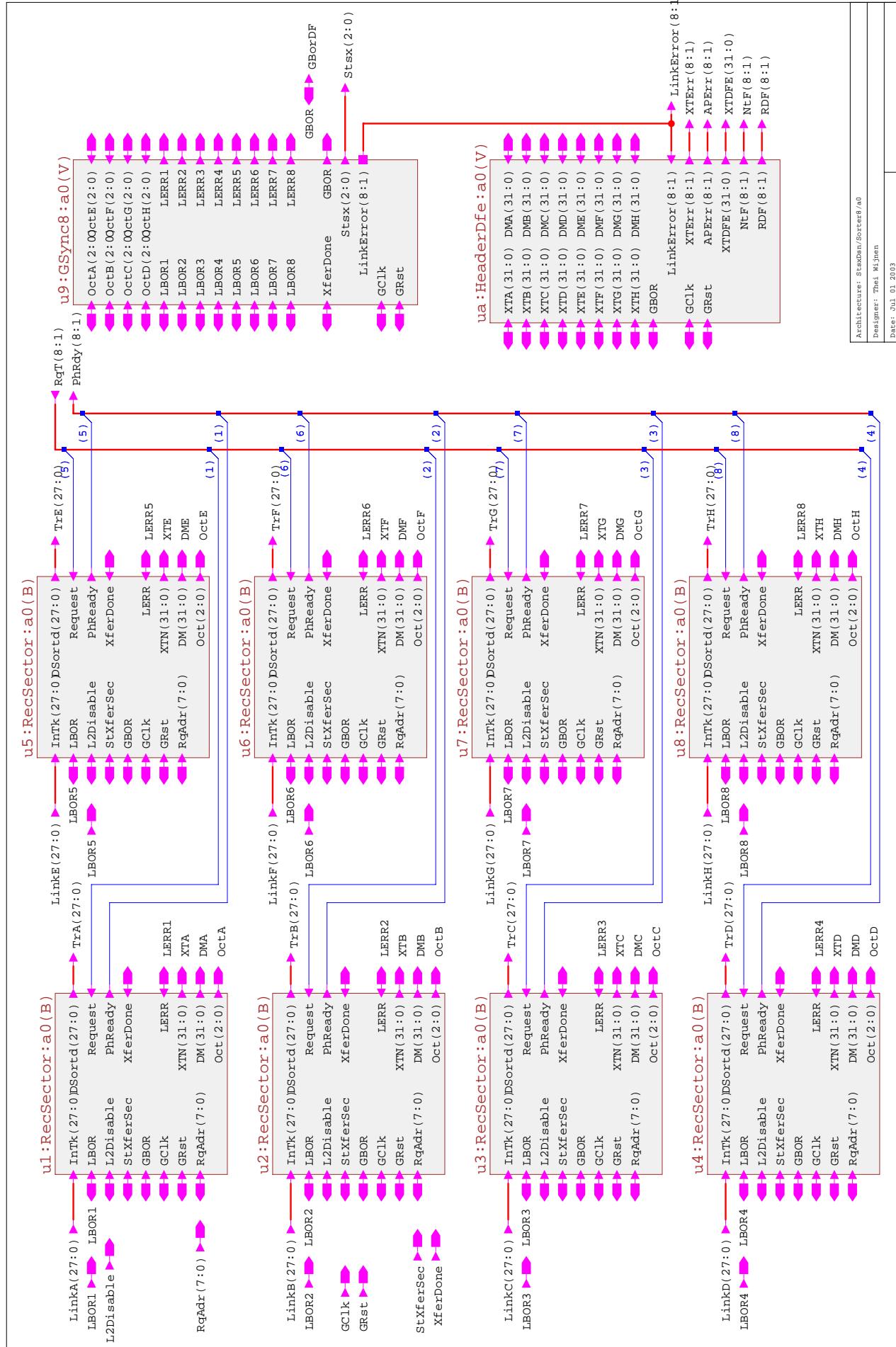
- |         |  |
|---------|--|
| Capture | (V) This entity maps the incoming number of objects for each link into an array and sends out the token to activate the header, data and trailer generators that send out the data.  |
| HGen    | (V) This entity generates the two L3 header words, sending the total number of objects, some flags and the crossing and turn number.   |
| Mux     | (V) This entity multiplexes the header words, the trailer and the data words before handing them to the GLink multiplexer and register (GMux).   |
| DGen    | (B) This block entity addresses the dual port memories and gets the L2CFT input records and sends them to the multiplexer (Mux). It consists of: <ul style="list-style-type: none"><li>– addr_gen (V) This entity controls the address lines and the selection of each of the dual port memories.</li><li>– link_ctr (V) This entity counts the number of objects that “addr_gen” is extracting from each of the dual port memories. When the memory of one input link is done, it passes control to the next memory.</li><li>– send_data (V) This entity divides the 32 bit data from the memory in two 16 bit frames.</li><li>– pctl (V) This entity is a simple token register that controls and passes the token to the next entity in the ring of entities “send-header”, “send-data” and “send-trailer”.</li></ul> |

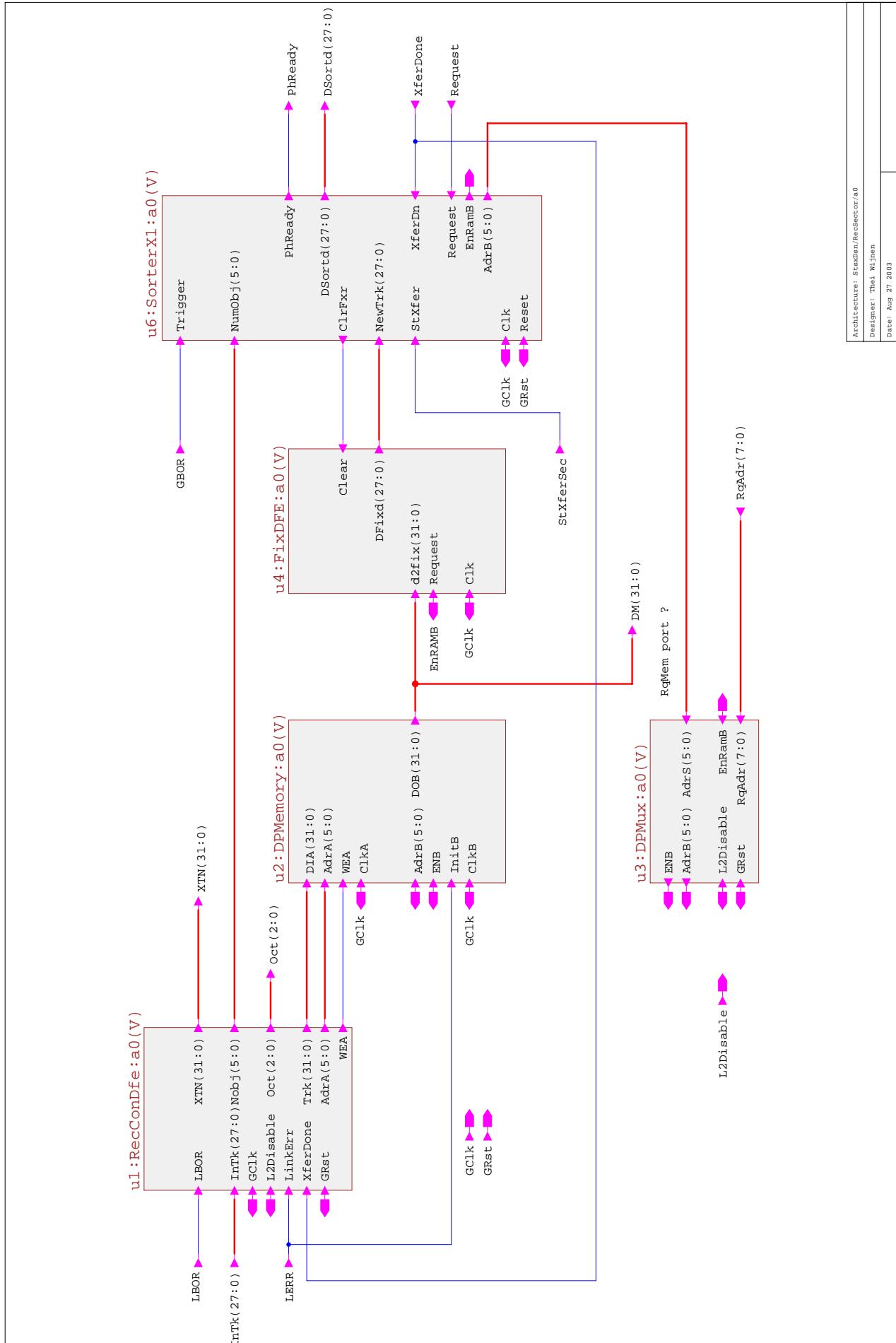
## 10 STSX Design schematics

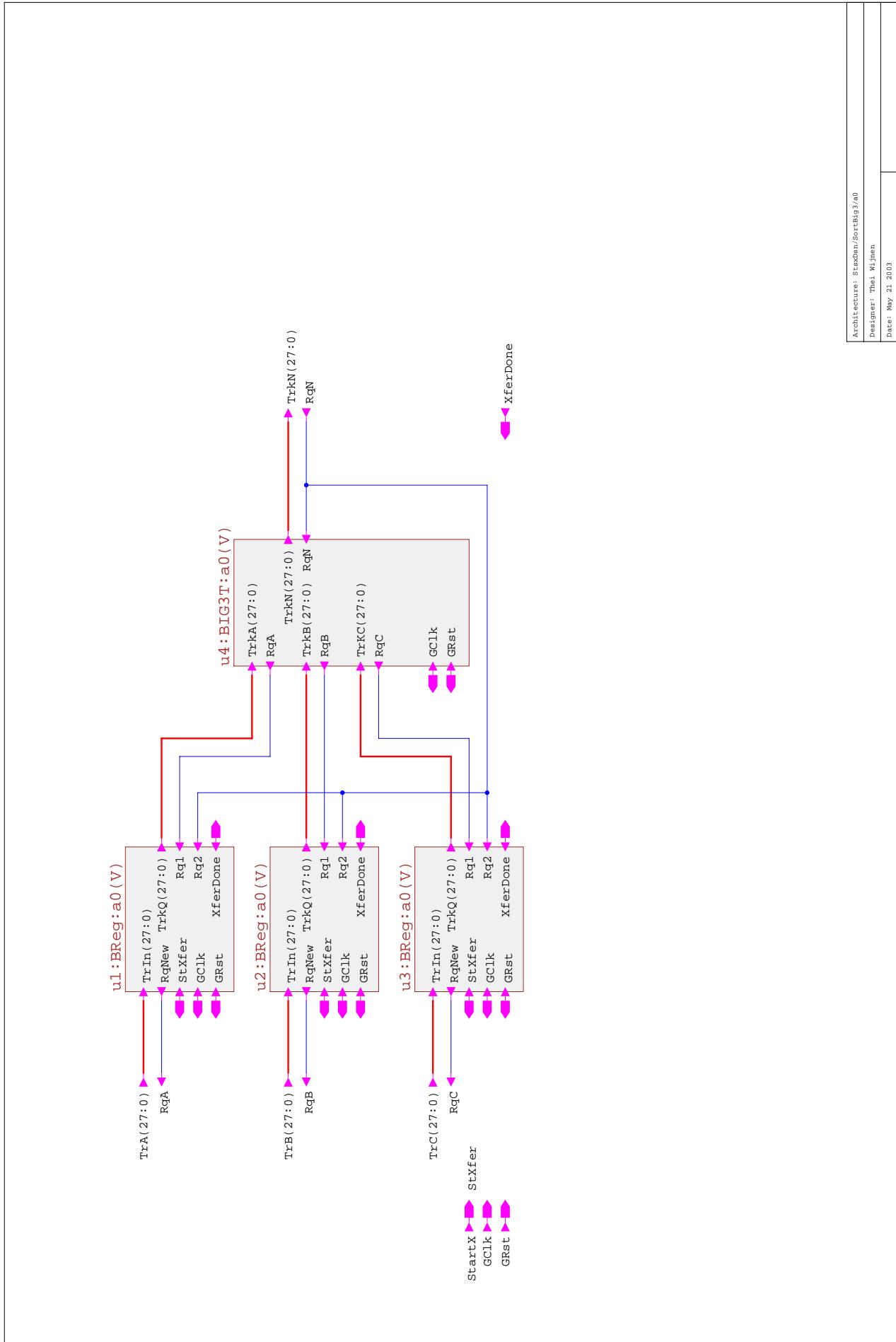
On the following pages the schematics for each of the block entities in the STSX design are shown.

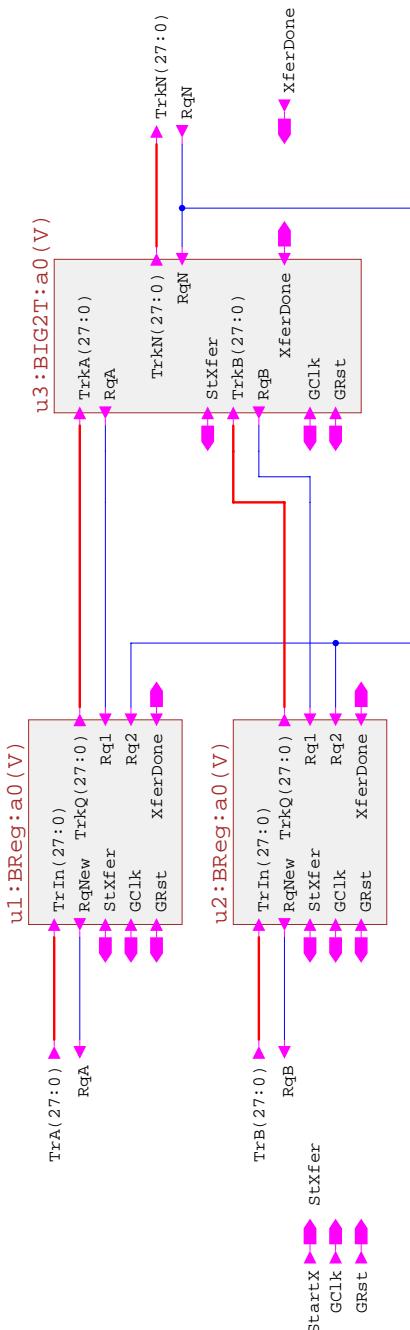




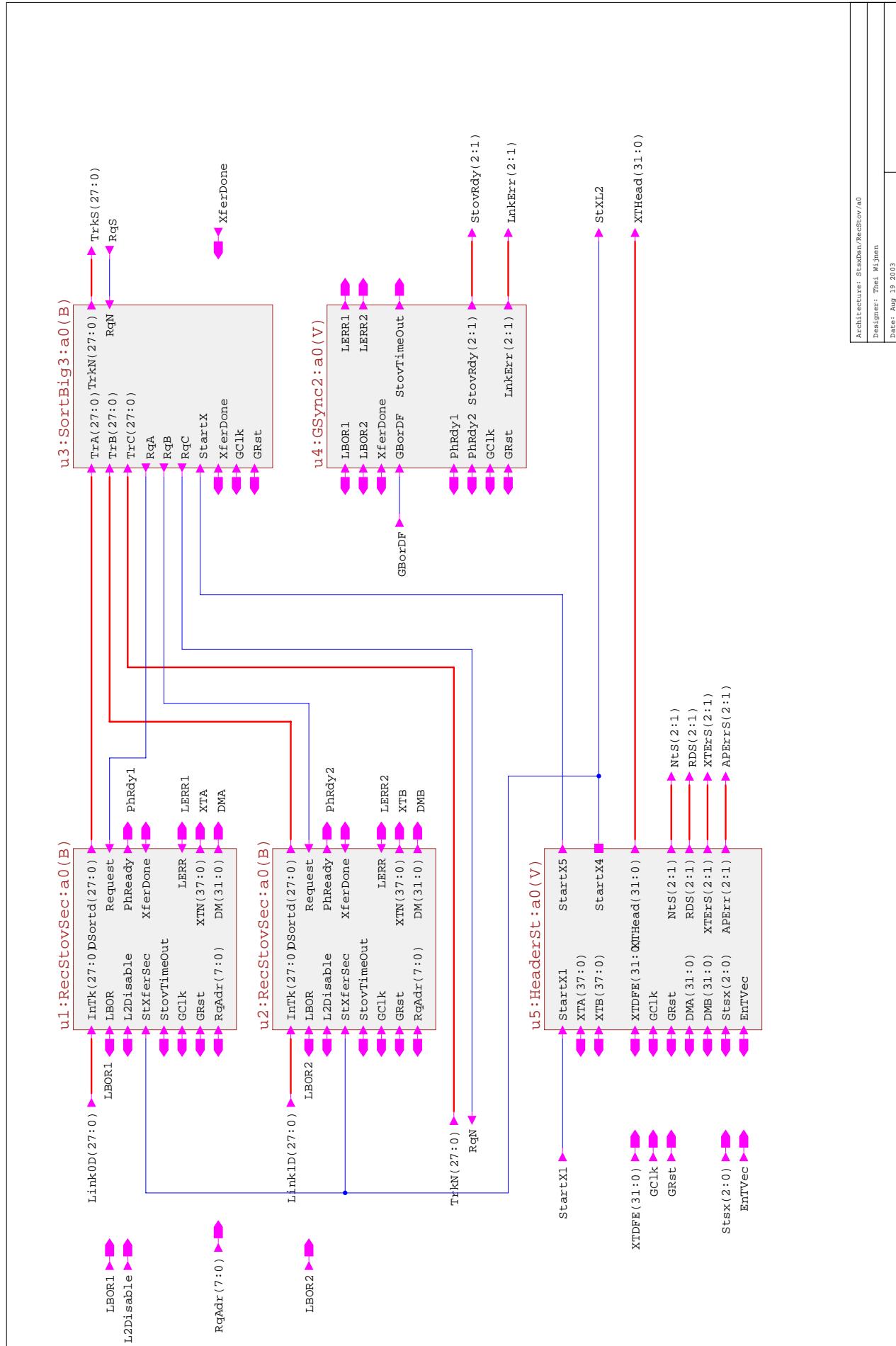


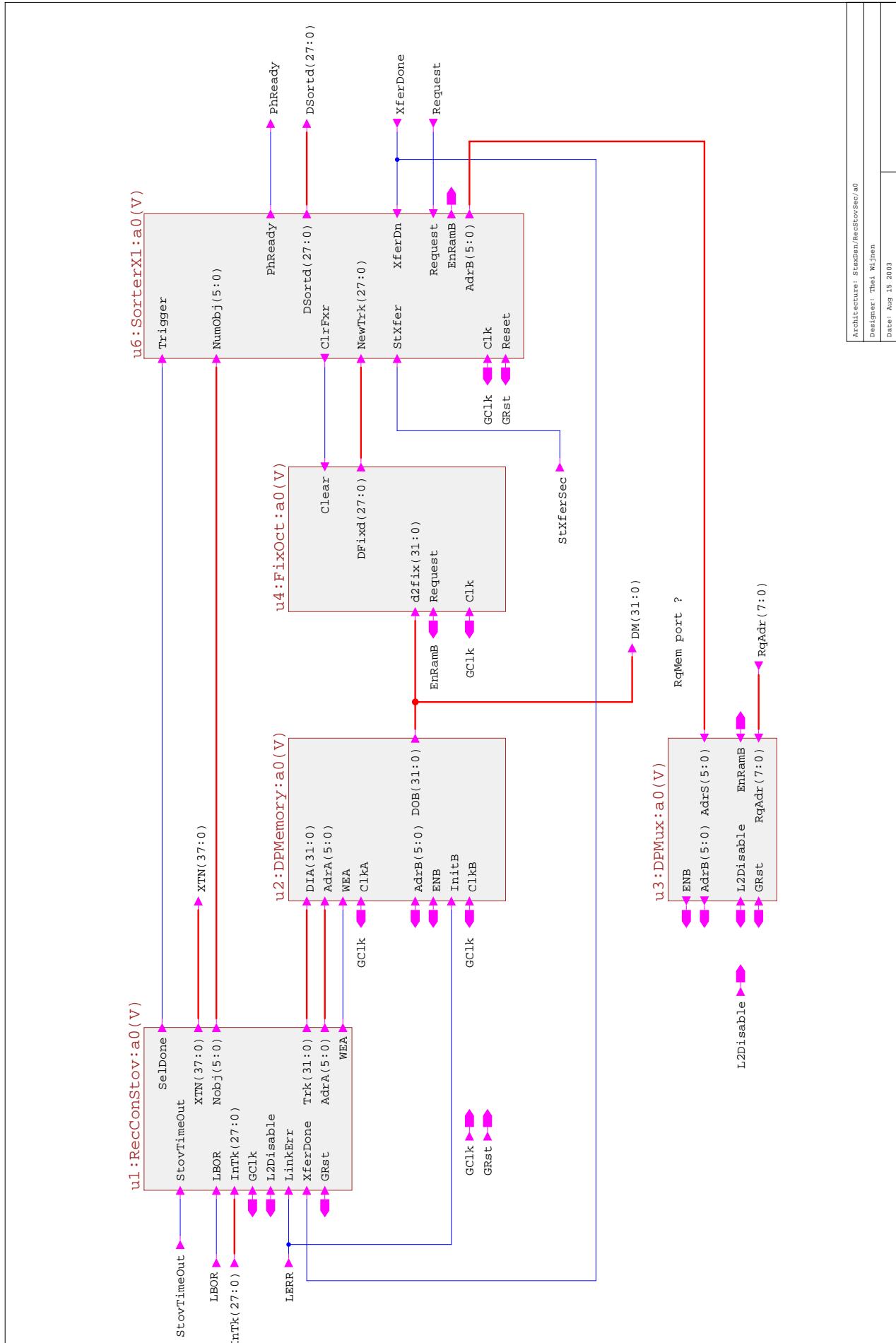


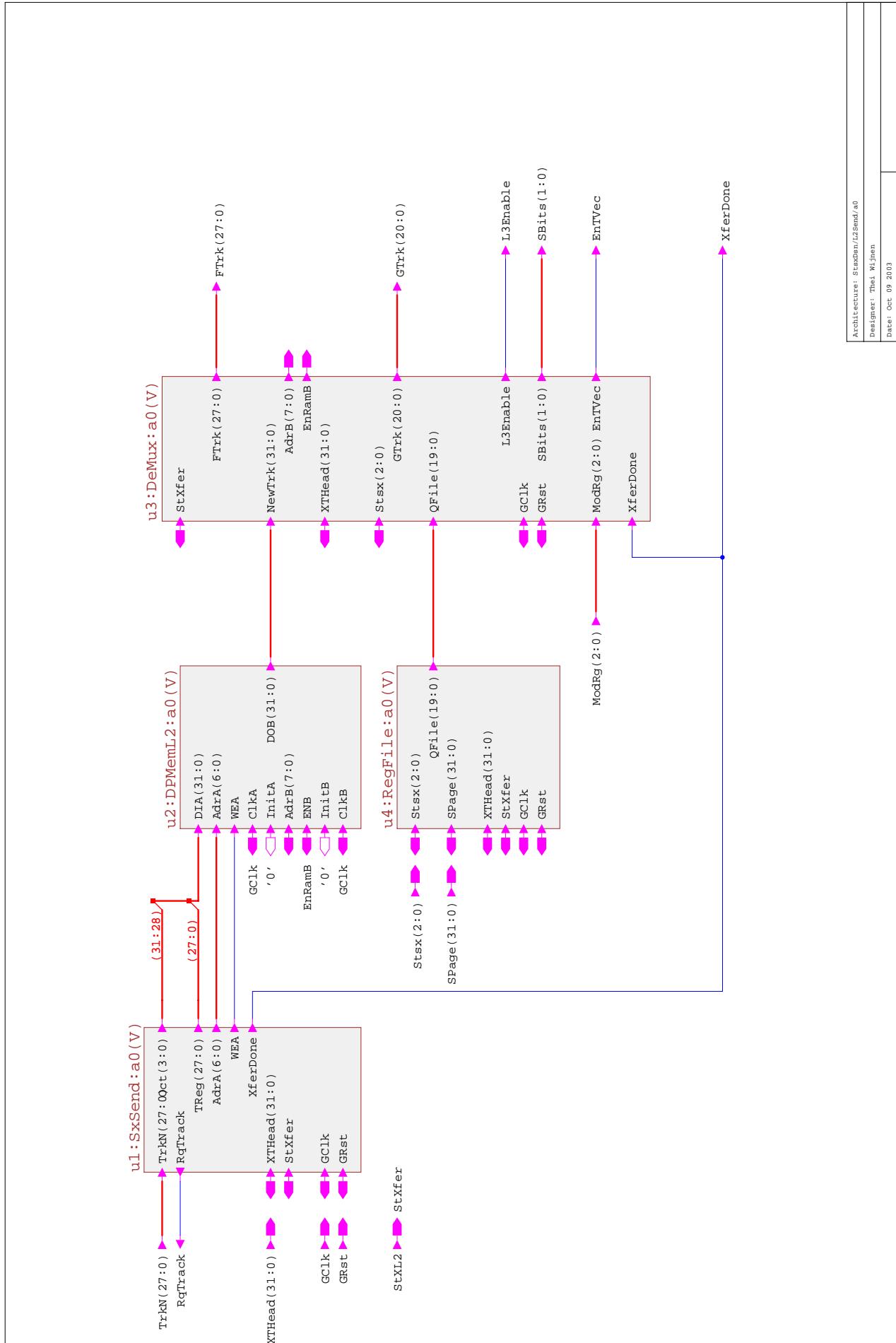




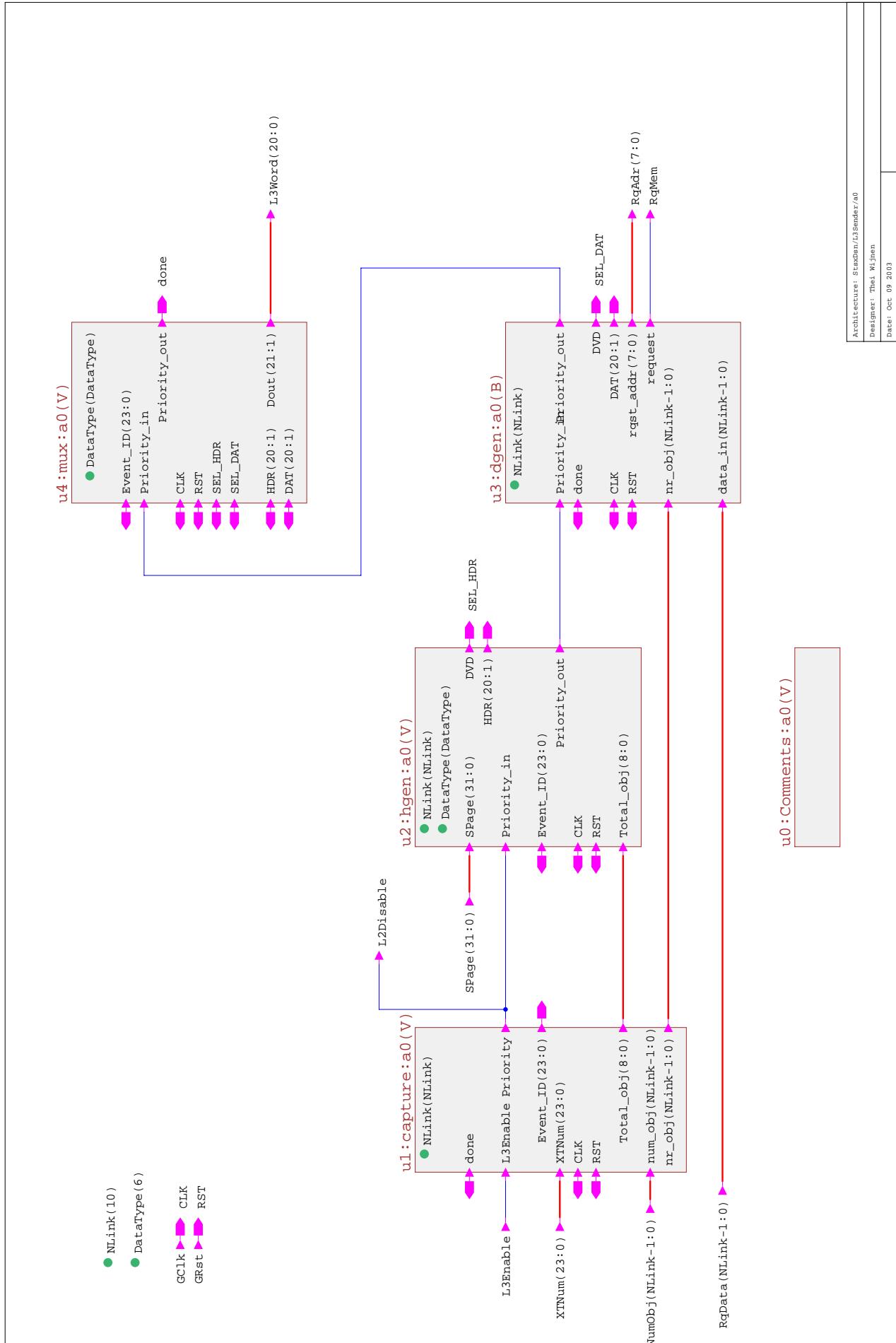
Architecture: Stuttgart/BIG2/A0
Designer: Theo Wijnen
Date: Jan 28 2003

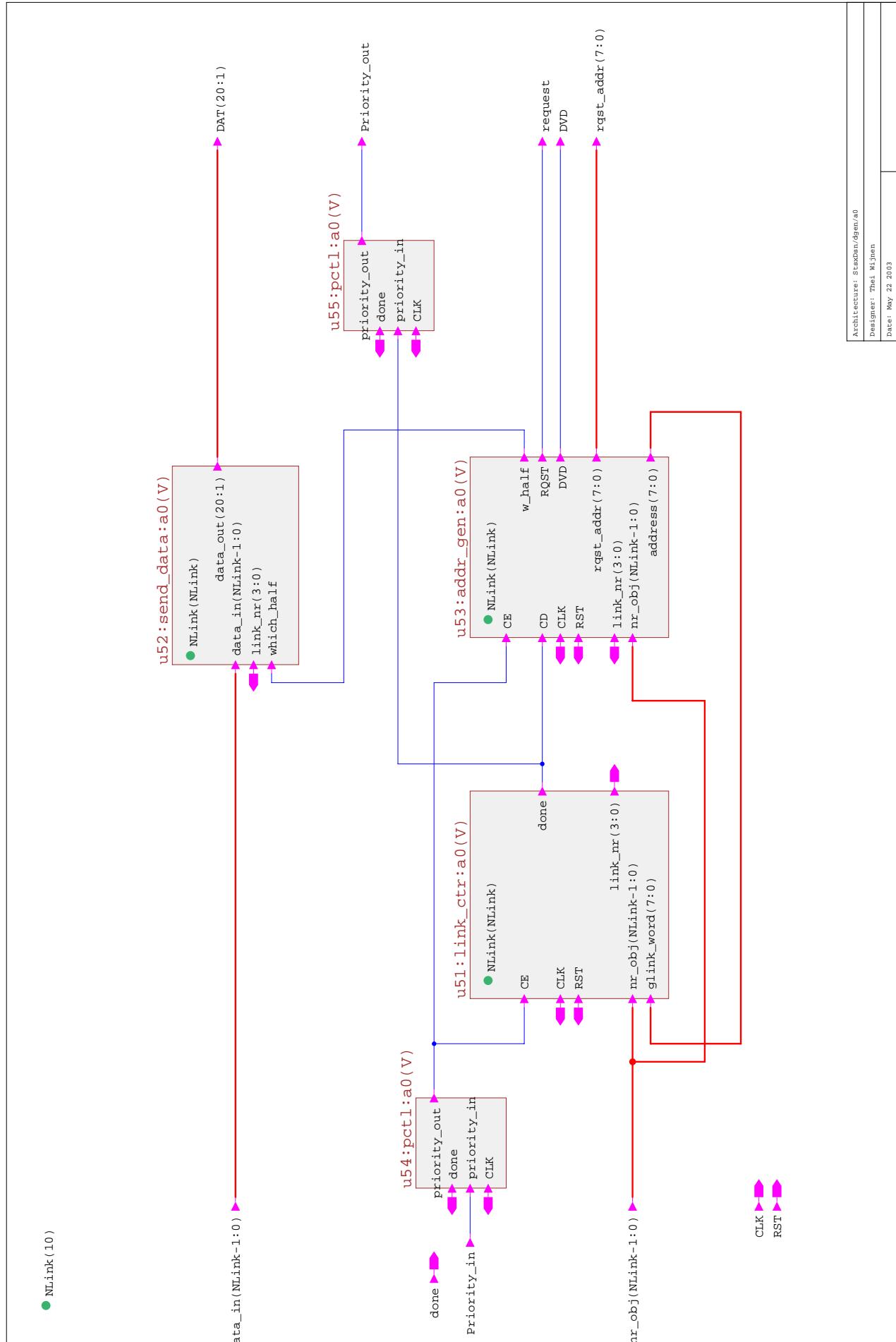


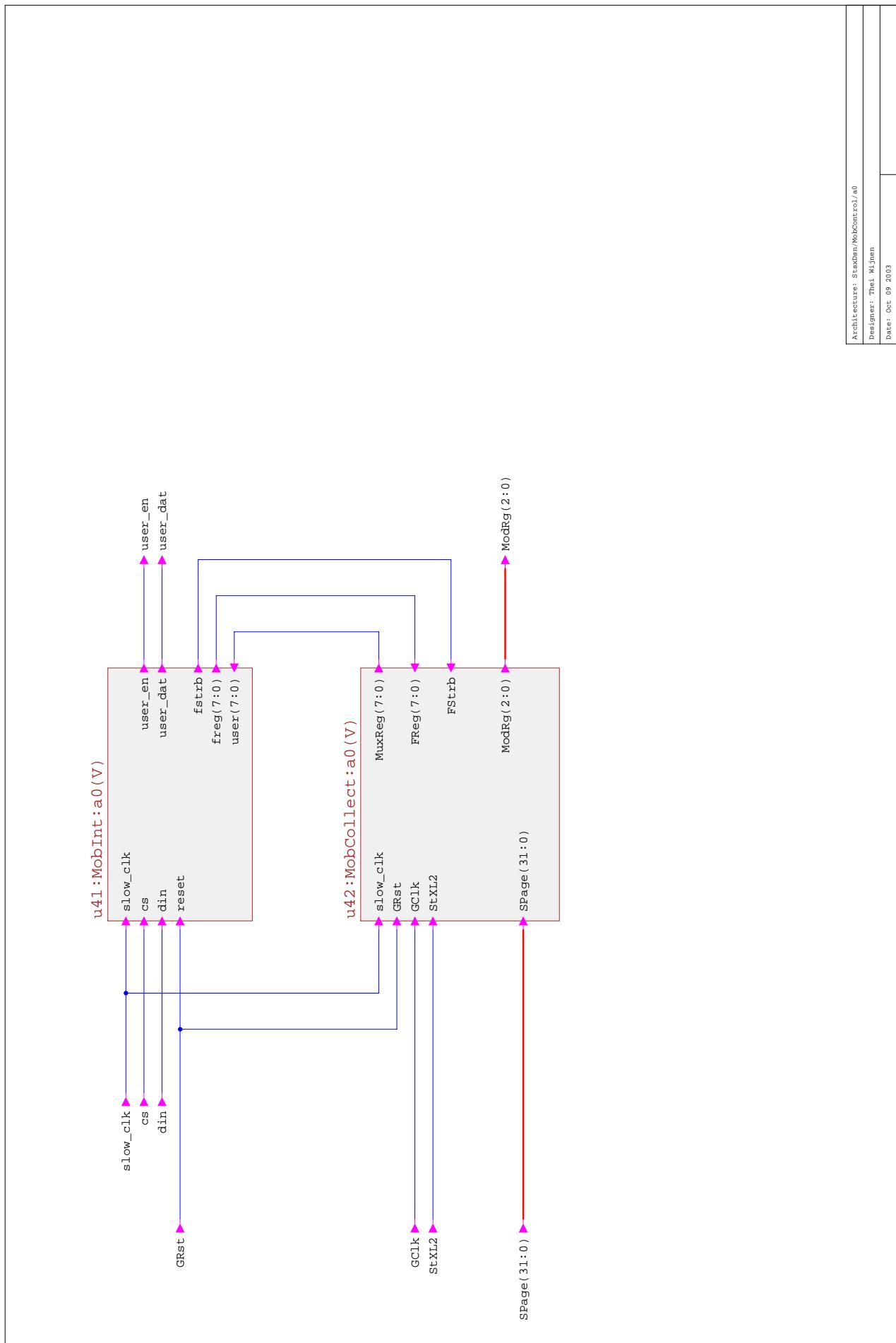




Architecture: STOV/2Send/40
Designer: Theo Wijnen
Date: Oct 09 2003







Architecture: STSxDn/MobControl/a0
Designer: Theo Wijnen
Date: Oct 09 2003

## 11 References